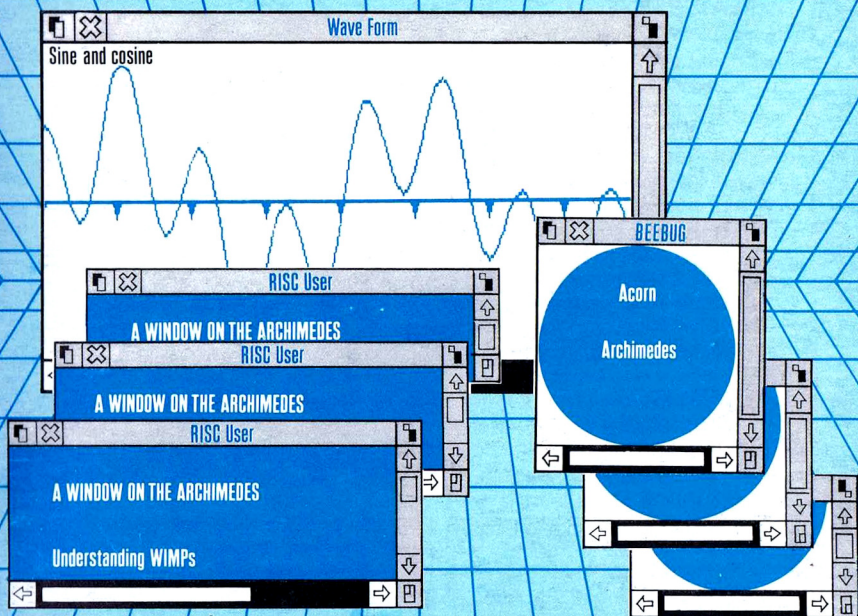


Volume 1
Issue 6

May
1988

RISC USER



A window on the Archimedes

THE MAGAZINE AND SUPPORT GROUP
EXCLUSIVELY FOR USERS OF THE ARCHIMEDES

CONTENTS

FEATURES

News	4
3D Graphics (Part 2)	7
Basic V - Return to Sender	12
Archimedes Visuals	18
A Window on the Archimedes (Part 2)	20
Introducing ARM Assembler (Part 2)	25
Hints & Tips	34

UTILITIES

Fast Directory Copier	5
An Intelligent Auto-Configure	13
Screen Manager Update	22
VDU Planning Sheet Generator	30
A Module for Twin	33

REVIEWS

Alphabase	10
Architext	17
Video Genlock Podule	23
Support for Artisan	24
The Dread Dragon Droom	29

RISC User is published by BEEBUG Ltd.
Co-Editors: Mike Williams, Dr Lee Calcraft
Assistant Editor: Kristina Lucas
Technical Editor: David Spencer
Production Assistant: Yolanda Turuelo
Technical Assistant: Lance Allison
Subscriptions: Mandy Mileham

BEEBUG, Dolphin Place, Holywell Hill, St.Albans, Herts
AL1 1EX. Tel. St.Albans (0727) 40303

All rights reserved. No part of this publication may be reproduced without prior written permission of the Publisher. The Publisher cannot accept any responsibility whatsoever for errors in articles, programs, or advertisements published. The opinions expressed on the pages of this journal are those of the authors and do not necessarily represent those of the Publisher, BEEBUG Limited.

BEEBUG Ltd (c) 1988



**The Archimedes Magazine
and Support Group.**

EDITORIAL

Last month Acorn Computers announced an annual loss of £3.3M, compared with a profit the previous year. It is always bad news for computer users when the company who produced their machine hits hard times. In the short term it may mean that Acorn will be considering how to further reduce their overheads. And when this happens, both current support, and future projects become threatened. It is clearly in the interests of users that neither of these activities are badly affected. Acorn are known to be working on a new range of computers which make even better use of ARM technology than does the Archimedes. We fervently hope that this work will not be affected.

Of course it all comes down to volume of sales, and Acorn will not improve their financial position until they can sell large quantities of their new machines; and demand will not truly ramp up until there is perceived to be a wide range of software for the machine. In some respects it is the sheer excellence of the machine that has worked against Acorn here. Many software houses, awed by the vast potential offered by the Archimedes, have opted to create products which make full use of this potential, and which, as a consequence, involve a considerable development time. Not until these products reach the marketplace will the Archimedes achieve the software support which it warrants, and which cautious would-be purchasers are waiting for.

This month's magazine disc is again packed full of goodies. In addition to all the programs from the magazine, it contains part two of Acorn's stunning 3D animation and an upgraded version of the RS232 fix, together with sample screens from Droom which is reviewed in this issue.

**We hope to see you on the
RISC User stand at the
Micro User Show (May 13 to 15).**

RISC CHIPS SECOND SOURCED

Acorn Computers has just announced that the ARM chip set at the heart of the Archimedes is also to be made in Japan by the electronics giant Sanyo. The ARM chip set is already manufactured by VLSI Technology of California. If experience with other new chips is anything to go by, then this 'second sourcing' will very rapidly lead to price reductions in the chip cost, and subsequently to the adoption of the ARM processor for other computers. Acorn particularly want to see the chip set being used for control applications, such as production line controllers.

NEW PC-EMULATOR

Acorn has released yet another version of the PC-Emulator for the Archimedes series. Version 1.20 claims to be faster than the older 1.09 release, and will also run additional software. According to Acorn, the new emulator, which uses MS-DOS 3.21, will run such packages as Symphony, dBase III, SideKick and Supercalc.

Users can upgrade to the new version by returning their original discs with a cheque for £15 to Customer Services, Acorn Computers, Fulbourn Road, Cherry Hinton, Cambridge CB1 4JN. Alternatively, RISC User and BEEBUG members can exchange their discs for just £12, provided they quote their membership number, by sending a cheque and the old discs to BEEBUG at the address on the inside front cover.

SHAREWARE

Context Computing is starting a scheme to market shareware to run under the PC-Emulator on the Archimedes. Shareware is a system by which quality software is distributed for very little cost, and can be freely passed on to other users. The documentation for shareware is normally supplied in the form of a text file along with the program, but sometimes full information can be obtained at a nominal cost.

Most existing PC shareware is on 5.25 inch discs, and not all of it will run using the Archimedes PC-Emulator. Context Computing is aiming to collect and test shareware, and then distribute it on 3.5 inch Archimedes discs. The first disc available is a full-feature spreadsheet

at £7.50, with other titles promised in the near future. Context Computing can be reached at 15 Woodlands Close, Cople, Bedford MK44 3UE, or by ringing (02303) 347.

ARCHIMEDES 440

Despite Acorn's previously announced plans, the Archimedes 440 is still in very short supply. Acorn now says that existing orders from dealers will be supplied in June, while anybody placing a new order will have to wait until September for delivery. There is also no definite date for the release of the Archimedes 410, but Acorn has confirmed that this machine will be released, probably later this year.

ARCHIMEDES PIPEDREAM

Pipedream, the word processor - cum - spreadsheet - cum - database from Colton software, is soon to be released for the Archimedes. Pipedream is currently supplied as standard on Cambridge Computer's Z88 portable computer, and is also available for the IBM PC and compatibles, or in the form of View Professional for earlier BBC micros.

The Archimedes version of Pipedream runs under the WIMP manager using pull-down menus, and includes features not available on the other versions, such as macros and Lotus file compatibility.

Archimedes Pipedream will be available from the middle of May at an inclusive price of £113.85. More details can be obtained from Robert Macmillan, Broadway House, 149-151 St Neots Road, Hardwick, Cambridge CB3 7QJ, or by ringing (0954) 211472.

GRAPHICS STUDIO

We have now received from Micro Studio a final version of the Archimedes graphics library, which we featured in last month's news. As well as over 100 pictures in the form of sprite files, the disc also contains some full-screen monochrome and colour pictures to load directly into Artisan.

RU

Note that, the price of the graphics library is £21.95, and not as stated last month.

FAST DIRECTORY COPIER

```

190 IF LEFT$(D$,1)="*" OSCLI D$
200 UNTIL LEFT$(D$,1)<>"*"
210 IF LEFT$(S$,INSTR(S$,".")-1)=LEFT$
(D$,INSTR(D$,".")-1) AND INSTR(LEFT$(S$,
INSTR(S$,".")-1),":")<>0 samedisk%=TRUE
ELSE samedisk%=FALSE
220 DIM space% -1
230 space%=HIMEM-space%&2000
240 DIM workbuffer% space%
250 canuse%=workbuffer%
260 PROCwaitondisk("Source")
270 PRINT"Reading directory tree"
280 PROCcreaddir(S$,0)
290 PROCwaitondisk("Destination")
300 PROCcreatedir(D$,workbuffer%)
310 PROCwaitondisk("Source")
320 workstart%=canuse%
330 workend%=workbuffer%+space%
340 filecount%=0
350 PROCreadin(S$,workbuffer%)
360 PROCbashout
370 END
380 :
390 DEFPROCcreaddir(dir$,parent%)
400 LOCAL ourspace%,offset%,entry%,ent
ryoff%
410 ourspace%=canuse%
420 ourspace%!4=parent%
430 canuse%+=8
440 offset%=0
450 REPEAT
460 SYS "OS_GBPB",10,dir$,canuse%,1,of
fset%,space%,"*" TO ,,read%,offset%
470 canuse%+=32
480 UNTIL offset%=-1 OR read%=0
490 IF read%=0 canuse%-=32
500 !ourspace%=(canuse%-ourspace%-4)DI
V 32
510 IF !ourspace%<>0 THEN
520 FOR entry%=0 TO !ourspace%-1
530 entryoff%=ourspace%+8+32*entry%
540 IF entryoff%!&10=2 THEN
550 entryoff%!&10=canuse%
560 PROCcreaddir(dir$+"."+FNgetzerostri
ng(entryoff%+&14),ourspace%)
570 ELSE
580 entryoff%!&10=0
590 ENDIF
600 NEXT
610 ENDIF
620 ENDPROC

630 :
640 DEFNgetzerostring(x%)
650 LOCAL y$
660 WHILE ?x%<>0
670 y$+=CHR$?x%
680 x%+=1
690 ENDWHILE
700 =y$
710 :
720 DEFPROCcreatedir(dir$,data%)
730 PRINT "Checking directory ";dir$
740 LOCAL entry%,entryoff%
750 SYS "OS_File",8,dir$
760 IF !data%<>0 THEN
770 FOR entry%=0 TO !data%-1
780 entryoff%=data%+8+32*entry%
790 name$=dir$+"."+FNgetzerostring(ent
ryoff%+&14)
800 IF entryoff%!&10=0 THEN
810 ELSE
820 PROCcreatedir(name$,entryoff%!&10)
830 ENDIF
840 NEXT
850 ENDIF
860 ENDPROC
870 :
880 DEFPROCwaitondisk(A$)
890 IF samedisk% THEN
900 SYS "OS_Byte",15,1
910 PRINT"Insert ";A$," disc, and pres
s <Space>"
920 REPEAT UNTIL GET=32
930 ENDIF
940 ENDPROC
950 :
960 DEFPROCreadin(dir$,data%)
970 LOCAL entry%,entryoff%,name$
980 IF !data%<>0 THEN
990 FOR entry%=0 TO !data%-1
1000 entryoff%=data%+8+32*entry%
1010 name$=dir$+"."+FNgetzerostring(ent
ryoff%+&14)
1020 IF entryoff%!&10=0 THEN
1030 savename$=D$+MID$(name$, (LEN S$)+1
)
1040 newthrough%=0
1050 REPEAT
1060 throughfile%=newthrough%
1070 IF workend%-canuse%<&100 PROCbasho
ut:PROCwaitondisk("Source")

```

Continued on page 16



3D GRAPHICS

(Part 2)

In this follow up to last month's article, Mark Davis introduces hidden line removal, and also translates the program into assembly code for extra speed.

Having covered in the previous article the creation and display of three-dimensional wire frame objects in Basic, I shall take two further steps this month. The first is to make use of ARM assembly code to increase the speed at which the shapes can be drawn; the second is to employ some form of hidden line removal to achieve greater realism. For this, a new data structure is necessary in the program, the purpose of which is to handle surfaces rather



than just the connection of lines.

As in the last program, the first set of DATA statements pro-



vide the co-ordinates used in the shape. But the second set arranged in groups of four, give the co-ordinates used to define the surfaces (each surface is defined by four points). To define a three-sided surface, the fourth point should equal the third. This can be seen in listing 1, where both three and four sided surfaces are used.

The hidden line removal calculations are carried out by means of vector mathematics. (For a full explanation of this see: J.McGregor & A.Watt, *Advanced Programming Techniques for the BBC Micro*, or M.Morris, *Computer Graphics and CAD Fundamentals*). Essentially, these calculations check whether the normal of a surface (i.e. a line perpendicular to the surface) is at an angle of greater than 90 degrees to the viewpoint. If so, the surface cannot be seen.

To use the program type in listing 1 and save it. Once the program is run, you will be presented with a house shape that will roll about on all three axes. On pressing the Space bar, the rotation will stop, and the view of the house will then be controlled by the mouse, as

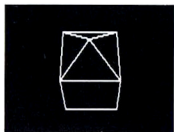
in last month's program. However, this time hidden lines are not displayed, giving the appearance of a solid object.

HOW THE PROGRAM WORKS

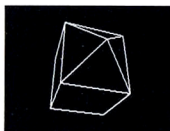
The program can be broken down into several stages:-

Lines 240-300 set up the coordinate and surface data, and use the variables 'coords' and 'surfs' (set up in line 90) as loop counters.

Lines 400-450 (PROC-



rotate) are used to rotate the shape, calling the machine code with R0-R2 containing the rotational parameters (xrot, yrot, zrot).



Lines 470-2090 set up the machine code in memory. This can be explained as follows:

Lines 570-1060 rotate each point around the origin, using registers



R0-R2 as parameters.

Lines 1080-1300 contain the necessary mathematics to determine whether a particular surface is visible or not (R0=surface no.).

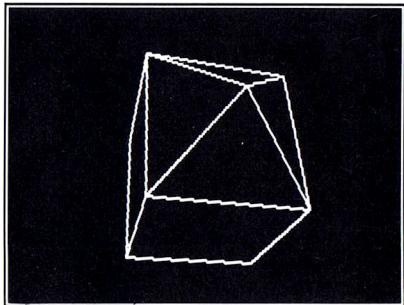
Lines 1550-1870 actually plot a surface on the screen, if it is visible (entered with R0=surface no.).

Lines 1890-1970 plot the whole shape on the screen, by calling lines 1550-1870 inside a loop.

Lines 2030-2080 set up a sine/cosine and reciprocal table in memory, to speed up the mathematics in the machine code.

Lines 2110-2330 provide a simple demonstration of the program, by rotating the shape defined in lines 310-380 around all 3 axes.

Incidentally, if you look closely at the program you will notice that before drawing the shape each time, it clears the screen. This works fine with mode 0, but can cause some flicker in higher modes. You can improve this



by using Exclusive-Or plotting to remove the old shape from the screen, rather than by clearing the whole screen.

CREATING YOUR OWN SHAPES

By taking a series of simple steps it is fairly easy to modify the program to use your own shapes. The method for doing this is as follows:

- 1) Determine how many separate points exist in your shape (the house shape in the program needed 9 points; a cube would need 8), set the value of 'coords' in line 90 to equal this, and then replace lines 310-380 with a list of these points, giving the X, Y, and Z coordinates of each.
- 2) Count the number of faces on your shape and place this number in line 90 as the value of 'surfs'.
- 3) Replace lines 360-380 with the new set of surface data. Each surface needs four values, stating which four points make up the outline of the face. With three-sided faces, make the fourth pair of values equal to the third. With faces of more than four sides, you must split the face into smaller shapes, and define each separately. This is rather like the technique on the Beeb of filling a complex area with a sequence of triangles.

```
10 REM >HiddDraw
20 REM Program 3D Drawing
30 REM Version A 1.0
40 REM Author Mark Davis
50 REM Risc User May 1988
60 REM Program Subject to copyright
70 :
```

```
80 DIM code $10000
90 coords=9:surfs=9
100 MODE0:ORIGIN 640,512:PROCass:CLS
110 xrot=0:yrot=-90:zrot=90
120 zpos=100:PROCinit:PROCdemo
130 *POINTER
140 MOUSE RECTANGLE -640,-512,1280,102
4
150 MOUSE TO yrot,xrot
160 REPEAT WAIT:CLS
170 MOUSE yrot,xrot,B
180 IF B AND 4 THEN zpos+=4*(zpos>32)
190 IF B AND 2 THEN zpos+=4
200 PROCrotate:IFI%<=0 THEN !zoff=zpos
*256:CALL draw
210 UNTIL FALSE
220 END
230 :
240 DEFPROCinit
250 FOR X=0 TO coords-1:READ x,y,z
260 ! (xw*X*4)=x*256:!(yw*X*4)=y*256
270 ! (zw*X*4)=z*256:NEXT
280 FOR X=0 TO surfs-1:FOR Y=0 TO 3
290 READ surf:?(surfdat*X*4+Y)=surf
300 NEXT:NEXT:ENDPROC
310 DATA 10,-10,-10,10,10,-10
320 DATA 10,10,10,10,-10,10
330 DATA -10,-10,-10,-10,10,-10
340 DATA -10,10,10,-10,-10,10
350 DATA 18,0,0
360 DATA 1,5,6,2,2,6,7,3,5,4,7,6
370 DATA 0,4,5,1,0,3,7,4,0,1,8,8
380 DATA 1,2,8,8,2,3,8,8,3,0,8,8
390 :
400 DEFPROCrotate
410 A%=(xrot+720) MOD 360
420 B%=(yrot+720) MOD 360
430 C%=(zrot+720) MOD 360
440 I%=USR rotate
450 ENDPROC
460 :
470 DEFPROCass
480 PRINT "Please wait..."
490 FOR PASS=0 TO 2 STEP 2
500 P%=code
510 [OPTPASS
520 .xa EQU0: .ya EQU0: .za EQU0
530 .xro EQU0: .yro EQU0: .zro EQU0
540 .xoff EQU0
550 .yoff EQU0
560 .zoff EQU0 80*256
570 .rotate
580 STMFED R13!,{R14}
590 MOV R12,#6:ADR R8,xw:ADR R9,sin
```




3D GRAPHICS

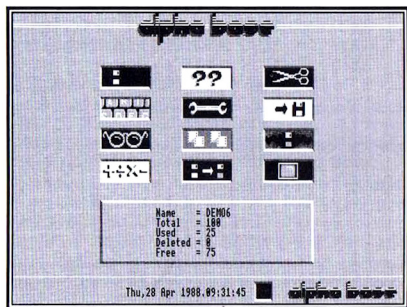
```
600 ADD R10,R9,#360:ADR R14,xr
610 MOV R0,R0,ASL#2:MOV R1,R1,ASL#2
620 MOV R2,R2,ASL#2:STR R0,xro
630 STR R1,yro:STR R2,zro
640 .rotlp
650 LDR R0,xro:LDR R1,yro:LDR R2,zro
660 LDR R3,[R8]:LDR R4,[R8,#yw-xw]
670 LDR R5,[R8,#zw-xw]
680 LDR R6,[R9,R2]:LDR R7,[R10,R2]
690 MUL R11,R3,R7:MOV R2,R11,ASR#8
700 MUL R11,R4,R6:SUB R2,R2,R11,ASR#8
710 STR R2,xa:MUL R11,R4,R7
720 MOV R2,R11,ASR#8
730 MUL R11,R3,R6:ADD R2,R2,R11,ASR#8
740 STR R2,ya:LDR R6,[R9,R1]
750 LDR R7,[R10,R1]:LDR R3,xa
760 LDR R4,ya:MUL R11,R3,R7
770 MOV R2,R11,ASR#8:MUL R11,R5,R6
780 SUB R2,R2,R11,ASR#8:STR R2,xa
790 MUL R11,R5,R7:MOV R2,R11,ASR#8
800 MUL R11,R3,R6:ADD R2,R2,R11,ASR#8
810 STR R2,za:LDR R6,[R9,R0]
820 LDR R7,[R10,R0]:LDR R3,xa
830 LDR R5,za:MUL R11,R4,R7
840 MOV R2,R11,ASR#8:MUL R11,R5,R6
850 SUB R2,R2,R11,ASR#8:STR R2,ya
860 MUL R11,R5,R7:MOV R2,R11,ASR#8
870 MUL R11,R4,R6:ADD R2,R2,R11,ASR#8
880 STR R2,za:LDR R3,xa:LDR R0,xoff
890 ADD R3,R3,R0:STR R3,[R14]
900 LDR R4,ya:LDR R0,yoff
910 ADD R4,R4,R0:STR R4,[R14,#yr-xr]
920 LDR R5,za:LDR R0,zoff
930 ADD R5,R5,R0:STR R5,[R14,#zr-xr]
940 CMP R5,#0:MOVL R0,#1
950 LDMLTFD R13!,{R15}:MOV R7,R14
960 BL recaddr:MOV R14,R7
970 BIC R5,R5,#&30
980 LDR R7,[R6,R5,LSR#4]:ADR R6,xs
990 MUL R11,R3,R7:MOV R11,R11,ASR#6
1000 STR R11,[R6,R12,ASL#2]!
1010 MUL R11,R4,R7:MOV R11,R11,ASR#6
1020 STR R11,[R6,#ys-xs]
1030 ADD R8,R8,#4:ADD R14,R14,#4
1040 ADD R12,R12,#1:CMP R12,#coords
1050 BCC rotlp:MOV R0,#0
1060 LDMLTFD R13!,{R15}
1070 :
1080 .hiddcheck
1090 STMPD R13!,{R14}
1100 ADR R4,surfdat:ADD R4,R4,R0,ASL#2
1110 LDRB R1,[R4]:LDRB R2,[R4,#1]
1120 LDRB R3,[R4,#2]:ADR R5,xs
1130 LDR R6,[R5,R1,ASL#2]
1140 MOV R6,R6,ASR#2
1150 LDR R7,[R5,R2,ASL#2]
1160 MOV R7,R7,ASR#2
1170 LDR R8,[R5,R3,ASL#2]
1180 MOV R8,R8,ASR#2:ADR R5,ys
1190 LDR R9,[R5,R1,ASL#2]
1200 MOV R9,R9,ASR#2
1210 LDR R10,[R5,R2,ASL#2]
1220 MOV R10,R10,ASR#2
1230 LDR R11,[R5,R3,ASL#2]
1240 MOV R11,R11,ASR#2
1250 SUB R0,R7,R6:SUB R1,R11,R9
1260 MUL R2,R0,R1:MOV R2,R2,ASR#8
1270 SUB R0,R8,R6:SUB R1,R10,R9
1280 MUL R3,R0,R1:MOV R3,R3,ASR#8
1290 SUB R0,R2,R3
1300 LDMPD R13!,{R15}
1310 :
1320 .xw EQU STRING$(coords*4,CHR$0)
1330 .yw EQU STRING$(coords*4,CHR$0)
1340 .zw EQU STRING$(coords*4,CHR$0)
1350 .xr EQU STRING$(coords*4,CHR$0)
1360 .yr EQU STRING$(coords*4,CHR$0)
1370 .zr EQU STRING$(coords*4,CHR$0)
1380 .xs EQU STRING$(coords*4,CHR$0)
1390 .ys EQU STRING$(coords*4,CHR$0)
1400 .surfdat
1410 EQU STRING$(surfs*4,CHR$0)
1420 .sin EQU STRING$(250,CHR$0)
1430 EQU STRING$(110,CHR$0)
1440 .cos EQU STRING$(250,CHR$0)
1450 EQU STRING$(250,CHR$0)
1460 EQU STRING$(250,CHR$0)
1470 EQU STRING$(250,CHR$0)
1480 EQU STRING$(220,CHR$0)
1490 EQU STRING$(220,CHR$0)
1500 :
1510 .xsa EQU xs
1520 .ysa EQU ys
1530 .surfdata EQU surfdat
1540 :
1550 .plot
1560 ALIGN
1570 STMPD R13!,{R14}
1580 MOV R12,R0:BL hiddcheck:CMP R0,#0
1590 LDMPD R13!,{R15}
1600 MOV R9,R12:LDR R0,surfdata
1610 ADD R12,R0,R9,ASL#2:LDR R10,xsa
1620 LDR R11,ysa:LDRB R0,[R12,#0]
1630 LDR R1,[R10,R0,ASL#2]
1640 MOV R1,R1,ASR#8
1650 LDR R2,[R11,R0,ASL#2]
1660 MOV R2,R2,ASR#8:MOV R0,#4:SWI &45
```

Continue on page 28

Mike Williams presents an early preview of Alphabase, Clares' database for the Archimedes.

Clares Micro Supplies is developing quite a name for itself with its range of applications packages for the Archimedes. The latest product is Alphabase, a card index style of database retailing at £49.95 now nearing completion.

It must be stated at the outset that we were again privileged with an early pre-release version of the software and accompanying documentation. This review is therefore very much an overview of the principal features and facilities of the package, and some details may change before the final version is released.



Once the disc is booted, Alphabase's very smart looking main menu is displayed on the screen. There are twelve quite large icons providing the main option choices, an information panel giving details of the current data file (if any), and at the foot of the screen a display of the current date and time (continually updated). The display is very clear, and in my view a distinct improvement on some of the menu screens employed in earlier software from Clares. Moreover, if you are one of those (like me) who takes a somewhat jaundiced view of the wealth of colourful but often puzzling icons used by Archimedes software, Alphabase allows all the graphics icons to be replaced by simple text legends instead.

CREATING DATA FILES

With Alphabase, creating a new data file is a simple but pleasurable experience. Select the

appropriate icon and you are presented with a blank screen designated Page 1. Use the mouse pointer to determine the position for a field name, and then type it in (maximum 10 characters). Then select the field type (string, real, integer, date, and formula are available). For string fields you use the pointer to determine the field length on the screen; the other field types have standard defaults. A formula field is like a spreadsheet cell - you specify a formula involving the names of other fields and the formula field is calculated when it is to be displayed or printed.

Each record, which may consist of as many as 400 separate fields may be spread out over 16 screen pages. Fields once created may be moved around within a page, and field names etc may be edited later as required. Once the data file has been created the details may be saved to disc. At this stage you are told how many records you have space for, and you can cut this down to reflect your real needs. Files may also be encrypted using a password, and duplicate screen formats may also be saved, and later edited to create different screen layouts.

DATA ENTRY

Data entry is quite straightforward, and offers three separate options. Global allows values to be replicated across a group of records, Normal allows data to be entered into fields in any order, while Mask constrains data to be entered in the order in which the fields are defined (in the current screen format).

BROWSE MODE

This option allows you to look at your data records, and to delete or amend them. You can jump directly to any record but you must specify it by record number.

CALCULATIONS

Alphabase has a Calculate mode which can be applied within a record or to a whole file, for example to total the contents of a given field throughout a file. There are a number of special keywords which facilitate this, such as !SUM (to sum the contents of a field), !AVR (to average the contents of a field), and !MAX and !MIN (to find the maximum or minimum of a given field).

SEARCHING AND SORTING

These are two of the most powerful and most useful options provided by Alphabase. When searching you can limit the range of records that will be processed. You can also select any one of four kinds of search (on a specific field, on the whole record, or for a maximum or minimum value). The syntax for search strings allows various relational operators (> < = etc) and special keywords like INCLUDES and EXCLUDES together with logical AND and OR. The search function produces a so-called search list, and then moves automatically to the sort option screen.

Not only can Alphabase sort on up to three key fields, but by the use of offsets it can also sort on part fields. For example, if a field contains both a first name and a surname, offsets would allow a sort on either first name or surname alone. However, the use of part fields as sort keys is dependent upon data being organised in 'words'; you cannot, say, specify a particular set of characters within a field (or set of fields) as a sort key.

Sorting of search lists is entirely within RAM and certainly appears to be fast. Note that whole files can be converted into search lists for sorting purposes without going through the search option. Search lists, which are effectively lists of pointers may also be saved and loaded separately to data files, and a search list may also be saved as an index file which then allows the use of a fast search option for rapid record retrieval.

PRINT FORMATS

A major part of any worthwhile database system will be concerned with print formats, and Alphabase is no exception. There are three ways of handling print requirements. Field order allows you to specify the order in which fields are to be printed, and together with some special characters and a set of so-called printer codes (controlling features such as condensed, enlarged, underlined text), complete print formats may be created. Again offsets may be used to select words within fields.

A second option allows records to be printed out in the form in which they appear on the screen, but remember that any one data file may have several different screen layouts associated with it, and this in fact provides a

way of creating a variety of print formats which can also be saved and loaded as required. A whole-screen menu allows you to specify and control page layout (headers, margins, footers and the like). Finally, in the print option, label printing (for names and addresses or whatever) is specifically catered for.

CONCLUSIONS

At this stage I can only give some initial comments. Overall I am very impressed by the clarity and simplicity of the various menu screens used by Alphabase. Icons are used, but not to excess, and their meanings are generally not that difficult to remember. Many of the menu screens are in text only, and in my view all the better for it.

The process of file creation (and any subsequent changes) is well thought out and a delight to use. In general the facilities are much what one would expect of a card index style of database, and all seems to work well. Several menu screens do require you to remember and type in field names, not always easy when you have to be exact. It would be useful to be able to open a scrolling window to check on this. I would also have liked a facility to compose a print form on screen using the mouse, rather than as a kind of formula (yes, I know screen formats may be used for printing, but they still don't offer the flexibility I have seen elsewhere). Finally, I would also liked to have seen at least some simple linking between two data files to provide added flexibility and economy of storage.

Overall, and this is clearly a personal view, I am more impressed by Alphabase, despite a few limitations, than some of Clares' earlier Archimedes offerings, but maybe we are all learning how best to use this machine. I would certainly recommend anyone looking for an easy-to-use computerised card index to consider Alphabase. If you do want more sophistication then consider the pricier Flying Start (reviewed in Issue 5), but remember that at present it will only run under the PC emulator.

Alphabase, £49.95 inc. VAT
Clares Micro Supplies
98 Middlewich Road, Rudheath,
Northwich, Cheshire CW9 7DA.
Tel. (0606) 48511

Mike Williams investigates the enhanced parameter passing provided by Basic V for functions and procedures.

Extensive use of functions and procedures has almost become the hallmark of well written programs in BBC Basic despite a number of significant limitations. Many of these constraints have now been removed with the advent of Basic V on the Archimedes. This month we'll deal with the way in which procedures may now return values to the calling program.

In previous versions of BBC Basic, parameters could only be used to pass values to a procedure (or function). Now, parameters can also be used to return values as well, just like the single value returned by a function.

When a procedure (or function) is defined with individual string or numeric parameters, its formal parameters (those specified in the definition) are treated as local variables to the procedure. When the procedure is called, the values of the parameters in the call are copied into these formal (local) variables. Any subsequent changes to the contents of *these* variables should not affect the values of the variables passed as parameters to the procedure.

Consider, for example, a procedure to order the values of two variables (smallest first):

```
1000 DEF PROCOrder(A,B)
1010 IF A>B THEN SWAP A,B
1020 ENDPROC
```

If you add the program:

```
100 X=2:Y=1
110 PROCOrder(X,Y)
120 PRINT X,Y
130 END
```

you will find that the two values assigned to X and Y remain as set in line 100 (un-ordered), because their values will have been copied to A and B when the procedure is called, and it is the values of A and B which are re-ordered. However, change line 1000 to:

```
1000 DEF PROCOrder(RETURN A,RETURN B)
using the keyword RETURN, and the correct answer will result because the final values of A and B are now returned to the variables (X and Y) with which the procedure is called.
```

It is important to be clear about the effect of this. The call to the procedure PROCOrder made at line 110 results in the values of X and Y being changed *in the main program*. There is inherent in this a limitation on the use of this feature. Whereas in the past procedures might be called with either variables or values supplied as the parameters, RETURN parameters may only be replaced by *variables* when the procedure is called (it is obviously impossible for Basic to return values to constants), so be careful.

What if you want a procedure to return one or more values, but without changing the contents of the variables whose values are passed to the procedure in the first place? Let's re-write the re-order procedure to do this:

```
1100 DEF PROCOrder2(A,B,RETURN HIGH,
RETURN LOW))
1110 IF A>B THEN HIGH=A:LOW=B
ELSE HIGH=B:LOW=A
1120 ENDPROC
```

and called using:

```
100 X=2:Y=1:MAX=0:MIN=0
110 PROCOrder2(X,Y,MAX,MIN)
120 PRINT MIN,MAX
130 END
```

Because A and B are not defined with the keyword RETURN, the values of X and Y in the call to the procedure (line 110) will remain unchanged. The value of the larger of X and Y is returned using the variable MAX, and similarly the smaller using the variable MIN. Note that it is essential to assign some dummy values to MAX and MIN (line 100) before the procedure is called, otherwise the variables are undefined as far as Basic is concerned, and an error message will result. The dummy values used are quite immaterial, of course.

With due care, the keyword RETURN used with parameters will greatly enhance the power of procedures in BBC Basic.

Next month I will deal with the other important extension to parameter passing, namely the use of arrays.

AN INTELLIGENT AUTO-CONFIGURE

by Lee Calcrafft

Use this program as a BOOT file to automatically reconfigure your machine to suit the needs of a particular application. Auto-configuring is only performed when absolutely necessary; and a special option activates a display of the machine's current state.

You will probably be familiar with the notion of auto-configuring the Archimedes. A number of pieces of commercial software employ the technique to configure the host machine to suit the software concerned, and automatically reconfigure the machine to its original settings after the application has terminated.

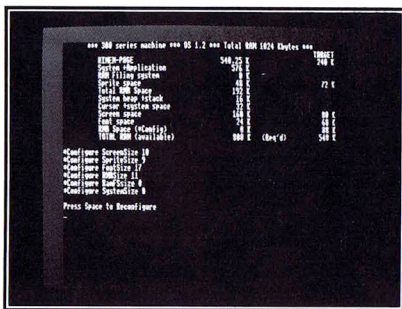
The accompanying program performs the same function, but has a number of enhancements. First of all, it will cater for the full range of machines, including those of the 400 series, whose larger "page" size is ignored by most auto-configure software. One major advantage of the program listed here is that it only reconfigures the host computer when this is essential. Most auto-configure software mindlessly configures *every* machine on which it runs (with consequent time overhead), regardless of whether it is necessary or not. A further strength of the accompanying program is that it contains an option to display the type and current state of the host machine, together with the target state. This is very useful for debugging purposes.

USING THE AUTO-CONFIGURE ROUTINE

When experimenting with auto-configuring software, it is very easy to lose your machine's current settings, so as a precaution, it would be wise to save these away using the Configuration Save routine from last month's RISC User Hints, or the CMOS RAM Manager from RISC User Issue 2. To test out the program, first type it in and save it to disc. Next, set your disc to auto-boot, using *OPT4,2 then alter the first few lines of the auto-configure program to meet your current requirements. Lines 140 to 180 should contain the amounts of RAM in kilobytes required by your application for each purpose e.g. a mode 15 screen will require 160K, so set **Tscreen** to 160, and so on. This is also the case for the RMA allocation: you should ignore any RMA RAM required by resident modules etc. When you have done this, set line 190 to give the name of

the program to be chained in by the configuring routine.

Now save the amended program under the name **IBOOT**, in the root directory of your disc; and if you press Shift-Break, it should all happen. Because we have left line 210 as **disp=TRUE**, the program will produce a display of the machine's state, and wait for the space bar to be pressed (change to **disp=FALSE** to omit this facility). It will then, if necessary, reconfigure the machine as prescribed, before chaining the application. After you have finished with the application, pressing Ctrl-Break (or performing a power-on reset) will restore your original configuration.



One further note: the program saves the host machine's configurations in a file called !CmosData in the root directory. It has exactly the same format as files used with either the CMOS RAM Manager or the Configuration Save routine. The loader automatically deletes this file when it terminates. You should note that the loader uses the *presence* of this file to indicate whether it is in the middle of reconfiguring (after performing an automatic hard Break), or whether it is being freshly run. When debugging the routine you should therefore always check that the root directory of

AN INTELLIGENT AUTO-CONFIGURE

your disc does *not* hold this file - erase it if you find it, and re-boot the loader.

I am grateful to Clares Micro Supplies for permission to use their hard Break routine employed in this program.

```

10 REM >RUconfig5
20 REM Program Auto-Configure
30 REM Version A 0.5
40 REM Author Lee Calcraft
50 REM RISC User May 1988
60 REM Program Subject to Copyright
70 :
80 DIM B% &20,code 40
90 DIM alloc(10),con$(6)
100 tot=0:K=1024
110 REM-----
120 REM Set requirements here
130 REM in Kbytes
140 Tbasic=233 :REM Basic
150 Tscreen=80 :REM Screen
160 Tsprites=70 :REM Sprites
170 Tfont=67 :REM Font
180 Trma=81 :REM RMA
190 name$="MainProg" :REM prog name
200 REM-----
210 disp=TRUE :REM Display ?
220 REM-----
230 IF FNostest<1.2 THEN PRINT"Operati
ng system 1.2 or above required":VDU7:EN
D
240 SYS "OS_File",17,":0.$.!CmosData"
TO file%
250 IF file%<>0 THEN
260 PROCreloadcmos
270 *DELETE :0.$.!CmosData
280 CHAIN name$
290 ENDIF
300 :
310 PROCTests:PROCbreakcode
320 IF disp THEN PROCdisplay
330 IF FNmet THEN
340 IF disp THEN PROCspace("Chain Prog
ram")
350 CHAIN name$
360 ENDIF
370 :
380 REM Are conditions attainable?
390 required=Tbasic+Tscreen+Tsprites+T
font+Trma
400 maxuseable=tot-alloc(5)-alloc(6)-a
lloc(4)+alloc(9)+alloc(8)
410 IF disp THEN
420 PRINT SPC10 "TOTAL RAM (available)
" TAB(33),maxuseable/K" K";
430 PRINT SPC3 "(Req'd)" TAB(64) requi
red/K" K""
440 ENDIF
450 IF required>maxuseable THEN PROCfa
ilmessage:END
460 :
470 PROCallsteal
480 IF disp THEN
490 FOR A=0 TO 5:PRINTcon$(A):NEXT
500 PROCspace("Reconfigure")
510 ENDIF
520 PROCsavecmos
530 FOR A=0 TO 5:OSCLI(con$(A)):NEXT
540 *CONFIGURE BOOT
550 *CONFIGURE DRIVE 0
560 CALL code
570 END
580 :
590 DEFPROCreloadcmos
600 CLOSE#0
610 handle%=OPENIN(":0.$.!CmosData")
620 FOR A=0 TO 239
630 OSCLI("FX162,"+STR$(A)+",""+STR$(BG
ET# handle%))
640 NEXT:CLOSE# handle%:ENDPROC
650 :
660 DEFPROCsavecmos
670 CLOSE#0
680 handle%=OPENOUT(":0.$.!CmosData")
690 FORA=0 TO 239
700 BPUT#handle%,FNcmos(A)
710 NEXT:CLOSE #handle%:ENDPROC
720 :
730 DEFPROCTests
740 IFFNfour THEN page%=&8000 ELSE pag
e%=&2000
750 PROCTvalueadjust
760 PROCreadram
770 alloc(7)=FNscrn
780 alloc(0)=HIMEM-PAGE
790 alloc(8)=&1000*FNcmos(134)
800 alloc(9)=page%*FNcmos(146)
810 FOR A=1 TO 7:tot+=alloc(A):NEXT
820 ENDPROC
830 :
840 DEFPROCfailmessage

```


AN INTELLIGENT AUTO-CONFIGURE

```

850 IF NOT disp THEN MODE0
860 VDU19,0,24,208,0,64
870 PRINT""This software requires mor
e than ";tot/K" K of RAM"
880 VDU7:ENDPROC
890 :
900 DEFFPROCspace(text$)
910 PRINT""Press Space to ";text$
920 REPEAT UNTIL GET=32
930 ENDPROC
940 :
950 DEFFPROCreadram
960 FOR A=1 TO 6
970 READ start%
980 start%=start%*10000
990 alloc(A)=FNrange(start%,page%)
1000 NEXT:ENDPROC
1010 :
1020 DEFFPROCcallsteal
1030 C$="*Configure "
1040 con$(0)=C$+"ScreenSize "+STR$(Tscr
een/page%)
1050 con$(1)=C$+"SpriteSize "+STR$(Tspr
ites/page%)
1060 con$(2)=C$+"FontSize "+STR$(Tfont/
(4*K))
1070 con$(3)=C$+"RMASize "+STR$(Trma/pa
ge%)
1080 con$(4)=C$+"RamFSSize 0"
1090 con$(5)=C$+"SystemSize 0"
1100 ENDPROC
1110 :
1120 DEFFPROCtvalueadjust
1130 IF Tscreen<40 THEN PRINT"SCREEN VA
LUE TOO LOW (40K Minimum)":VDU7:END
1140 Tbasic=FNroundup(Tbasic)
1150 Tscreen=FNroundup(Tscreen)
1160 Tsprites=FNroundup(Tsprites)
1170 Trma=FNroundup(Trma)
1180 Tfont=Tfont*K
1190 IF Tfont MOD (4*K) THEN Tfont+=4*K
-(Tfont MOD (4*K))
1200 ENDPROC
1210 :
1220 DEFFFNroundup(param)
1230 param=param*K
1240 =page%*((INT(param/page%)-((param
MOD page%)>0)))
1250 :
1260 REM Are conditions met?
1270 DEFFNmet

1280 Mbasic=(Tbasic<=alloc(0))
1290 Mscreen=(Tscreen<=alloc(7))
1300 Msprites=(Tsprites<=alloc(3))
1310 Mfont=(Tfont<=alloc(8))
1320 Mrma=(Trma<=alloc(9))
1330 =Mbasic AND Mscreen AND Msprites A
ND Mfont AND Mrma
1340 :
1350 DEFFNcmos(loc)
1360 SYS 6,161,loc TO X,Y,Z
1370 =Z
1380 :
1390 DEFFNrange(start%,page%)
1400 addr%=start%
1410 REPEAT
1420 Z%=FNcheck(addr%)
1430 addr%+=page%
1440 UNTIL Z%=FALSE
1450 =addr%-start%-page%
1460 :
1470 DEFFNcheck(addr%)
1480 SYS 3A,addr%,addr%+1 TO ;flag
1490 =(flag AND 2)=0
1500 :
1510 DEFFNtxt:RESTORE
1520 REPEAT READ start,text$
1530 UNTIL addr%<10000<=start:=text$
1540 DATA 0,&100,&140,&180,&1C0,&1F0
1550 :
1560 DEFFNscrn
1570 B%=(B%+3,DIV4)*4: !B%=150: ! (B%+4)=
-1
1580 SYS 31,B%,B%+10: ! (B%+10)
1590 :
1600 DEFFNfour
1610 SYS 1A ,0,0 TO reg
1620 =(reg AND 8)=8
1630 :
1640 DEFFNostest
1650 SYS 6+2^17,0,0 TO A%
1660 A$="":A%+=11
1670 FOR B=0 TO 3
1680 A$+=CHR$(A%?B)
1690 NEXT:=VAL(A$)
1700 :
1710 DEFFPROCdisplay
1720 MODE0:VDU19,0,24,128,128,240
1730 IF page%<2000 THEN A$="300 " ELSE
A$="400 "
1740 PRINT SPC7"*** ";A$"series machine
";

```

AN INTELLIGENT AUTO-CONFIGURE FAST DIRECTORY COPIER

```

1750 PRINT "*** OS ";FNostest;
1760 PRINT " *** Total RAM ";tot/K" Kbyt
es ****"
1770 FOR A=0 TO 9
1780 READ A$
1790 PRINTSPC(10)A$ TAB(33),alloc(A)/K"
K"
1800 NEXT
1810 PRINTTAB(70,1)"TARGET"
1820 PRINTTAB(64,2)Tbasic/K" K"
1830 PRINTTAB(64,5)Tsprites/K" K"
1840 PRINTTAB(64,9)Tscreen/K" K"
1850 PRINTTAB(64,10)Tfont/K" K"
1860 PRINTTAB(64,11)Trma/K" K"
1870 ENDPROC
1880 :
1890 DEFPROCbreakcode

1900 FOR I%=0 TO 9
1910 READ hex$
1920 !(code+I%*4)=EVAL("&"&hex$)
1930 NEXT
1940 ENDPROC
1950 :
1960 DATA E3A000C8,E3A01003,E3A02000,EF
020006,E3A0050E
1970 DATA E5901000,E7011001,EF020016,E3
3FF3FF,E3A0F000
1980 :
1990 DATA HIMEM-PAGE,System +Applicatio
n, RAM Filing system,Sprite space
2000 DATA Total RMA Space, System heap
+stack,Cursor +system space
2010 DATA Screen space,Font space,RMA S
pace (*Config)

```

RU

FAST DIRECTORY COPIER (continued from page 6)

```

1080 inhandle%=OPENIN name$
1090 !canuse%=entryoff%
1100 canuse%+=12
1110 filecount%+=1
1120 PRINT"Reading ";name$
1130 SYS "OS_GBPB",3,inhandle%,canuse%,
workend%-canuse%&100,throughfile% TO ,,
,notread%,newthrough%
1140 canuse%!-4=newthrough%
1150 canuse%!-8=throughfile%
1160 canuse%+=(newthrough%-throughfile%
+3)AND (NOT 3)
1170 $canuse%=savename$
1180 canuse%+=(LENsavename$+4)AND (NOT 3
)
1190 CLOSE#inhandle%
1200 UNTIL newthrough%=entryoff%!8
1210 ELSE
1220 PROCreadin(name$,entryoff%!&10)
1230 ENDIF
1240 NEXT
1250 ENDIF
1260 ENDPROC
1270 :
1280 DEFPROCbshout
1290 LOCAL entryoff%,i%
1300 IF filecount%>0 THEN
1310 PROCwaitondisk("Destination")
1320 canuse%=workstart%
1330 FOR i%=1 TO filecount%

1340 entryoff%!=canuse%
1350 startoff%=canuse%!4
1360 endoff%=canuse%!8
1370 canuse%+=12
1380 nameoff%=canuse%+((endoff%-startof
f%+3)AND (NOT 3))
1390 PRINT"Writing ";$nameoff%
1400 IF startoff%=0 AND endoff%=entryof
f%!8 THEN
1410 SYS "OS_File",0,nameoff%,entryoff%
!0,entryoff%!4,canuse%,canuse%+endoff%
1420 ELSE
1430 IF startoff%=0 outhandle%=OPENOUT
$nameoff% ELSE outhandle%=OPENUP $nameof
f%
1440 SYS "OS_GBPB",1,outhandle%,canuse%
,endoff%-startoff%,startoff%
1450 CLOSE#outhandle%
1460 ENDIF
1470 IF endoff%=entryoff%!8 SYS "OS_Fil
e",1,nameoff%,entryoff%!0,entryoff%!4,,e
ntryoff%!12
1480 canuse%=nameoff%+((LEN$nameoff%)+
4)AND (NOT 3))
1490 NEXT
1500 ENDIF
1510 canuse%=workstart%
1520 filecount%=0
1530 ENDPROC

```

RU



ARCHITEXT

David Spencer takes a look at Architext from Hopesoft, a new text editor offering both advanced features and ease of use.

Architext is a new mouse-controlled text editor, mainly for use in writing and editing programs in compiled languages such as C and Fortran. The software is supplied on a single disc, and our pre-release version was accompanied by a 20 page user guide.

Architext has to be configured before use, using a program supplied on the disc. One option allows you to select a version of the program that can be used to edit Basic programs, while another gives you the choice of mode 8 or 19, (both 80 characters per line, but mode 19, which is only available with a multisync monitor, gives 64 as opposed to 32 lines). There are two further options, those controlling search and replace, and the handling of Tab characters.

When the editor is first invoked you see a few red lines at the bottom of the screen and a rather badly designed cyan arrow which can be moved around with the mouse. The red lines define the control area of the screen, which consists of some arrow icons for scrolling, a scroll bar of sorts, and information on the file being edited.

Architext is controlled by a combination of the keyboard and the mouse, but this doesn't mean that it runs under WIMP manager, in fact far from it. The mouse is used to move a cyan arrow round the control area, or a carat round the text. The mouse's buttons can be used to move the cursor, select text, or bring up an editing menu.

One of the main features of Architext is the ability to load up to six files at one time. These extra documents can be displayed one at a time and edited as desired. It is also possible to copy and move from one to another. Unfortunately, this feature didn't work properly on my version, due to a problem with memory allocation which Hopesoft claim will be fixed in the final version.

Entering new text or editing existing text is fairly simple, although you are confined to insert mode at all times. The Tab key can be used to insert either a Tab character, or a certain number of spaces, depending on which is more appropriate. A further limitation is that you are restricted to the keyboard characters, which could be a problem if you are editing text containing special characters.

Architext, in common with other text editors, has fairly comprehensive search and replace facilities. It is possible to search and replace strings on a global or selective basis, and also to maintain search and replace buffers, so that text can be edited during a search, and the search then continued. Wildcard searches on a limited basis are allowed by using a dot to match any character, and a vertical bar to replace groups of characters. Sadly, none of the advanced pattern matching facilities found in Twin are present in Architext, and I feel this could make otherwise simple tasks very time-consuming and tedious.

Comparing Architext with Acorn's Twin, it is Architext which, in my opinion, gives the impression of being the back runner. However, Architext does allow effective use of the mouse, whilst Twin is confined to the keyboard, and at £19.90 Architext is over £13 cheaper than Twin, and thus fully justifies its price. Incidentally, for just £3.50 you can get a demonstration version of Architext, which will do everything but save to disc, and if you later decide to buy the full product your £3.50 is refunded. Despite this, I feel that Twin is altogether the more professional product, and the one I would choose.

RU

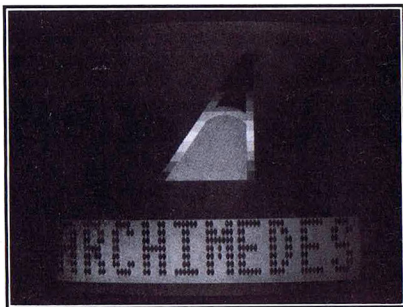
Product Supplier	Architext Text Editor Hopesoft Hope Cottage, Winterbourne, Newbury, Berkshire RG16 8BB. Tel. (0635) 248472
Price	£19.90 inclusive

ARCHIMEDES VISUALS

More startling visual effects by J.Boomgaardt and B.Christie.

THE SHINING ARC DISPLAY

The major Visual in this month's column comes from Jeroen Boomgaardt, and although somewhat longer than our usual listings, it is well worth the effort of typing it in. The program creates a large Archimedes "A" logo in the now familiar colours. But the body of the "A" appears to have a shining metallic surface whose reflection continually shifts, producing a quite breathtaking effect. At the same time, enlarged text scrolls across the lower part of the screen.



The text is held in DATA statements, and it is a trivial matter to alter this to suit your whim. The protocol is simple. The final data statement should contain a single space character (as on line 1040). The only other point is that you can reverse out the scrolled text by including character 164 at any point. A second occurrence cancels the effect. The control code 164, which can be entered directly from the keyboard with Shift-Pound (though not in Twin), is used in the example to reverse out the words "RISC User" in line 1020. Note that character 164 prints out differently on different printers. On an FX80, it appears as a \$ sign, and as a \$ in our listing.

The program uses a number of tricks that are well worth a mention. Firstly, it runs in mode 12, and achieves the shining effect by changing the palette of the white and grey colours which make up the body of the "A". This is performed in machine code (called regularly in line 380). The scrolling text is

handled in Basic, and uses OSWORD 10 to read the pixel definition of each character to be displayed. The routine which outputs the enlarged pixels makes use of a VDU call (VDU23,16,8) to send characters in a vertical sequence. To test this out, type:

```
VDU23,16,8|
```

in immediate mode, and see the effect when you type at the keyboard (to get things back to normal, use VDU23,16).

```
10 REM >ArcSpark3
20 REM Program   Shining Arc
30 REM Version   A 0.3
40 REM Author    Jeroen Boomgaardt
50 REM Risc User 1988
60 REM Program   Subject to Copyright
70 :
80 MODE 12:OFF
90 ON ERROR VDU23,16|:VDU26:ON:REPORT
:PRINT" at line ";ERL:END
100 DIM M% 9,param 5,code 150,rb 1,tb 1
110 ?rb=1:?tb=0
120 PROCassemble
130 PROCarch
140 VDU 19,0,24,0,0,240
150 VDU 19,0,4;0;19,2,3;0;
160 VDU 23,128,0,&1C,&3E,&7F,&7F,&7F,&
3E,&1C
170 VDU 28,0,30,79,23
180 VDU 31,79,0:VDU 23,16,8|
190 COLOUR 2:R%=0:V%=1
200 :
210 REPEAT
220 RESTORE
230 REPEAT
240 READ T$
250 FOR N%=1 TO LEN(T$)
260 C%=ASC(MID$(T$,N%,1))
270 IF C%=164 THEN VDU 23,17,5| ELSE P
ROCprint
280 NEXT
290 UNTIL T$=""
300 UNTIL FALSE
310 :
320 DEF PROCprint
330 ?M%=C%
340 SYS "OS Word",10,M%
350 P%=256
360 FOR K%=1 TO 8
370 P%=P%>>>1
380 WAIT:IF K% MOD 2 THEN CALL shine
390 FOR T%=1 TO 8
```

```

400 IF ((M%?T%)ANDP%)>0 THEN VDU 128 E
LSE VDU 32
410 NEXT:NEXT
420 ENDPROC
430 :
440 DEF PROCarch
450 VDU 24,440;440;1279;1023;
460 ORIGIN 440,440
470 GCOL 0,15
480 RECTANGLE FILL 375,0,30,490
490 MOVE 40,0:MOVE 405,490
500 PLOT &75,370,490
510 MOVE 285,130:MOVE 385,205
520 PLOT &A5,175,185:MOVE 285,130
530 MOVE 385,245:PLOT &A5,230,200
540 FILL 285,260
550 FOR Y%=0 TO 12
560 GCOL 2,Y%+3
570 RECTANGLE FILL 0,Y%*38,410,34
580 NEXT
590 GCOL 0,1:FILL 285,300
600 GCOL 0,2:FILL 285,130
610 ENDPROC
620 :
630 DEF PROCassemble
640 count=3:dir=4:col=5:tint=6
650 FOR pass=0 TO 2 STEP 2
660 P%=code
670 [ opt pass
680 .shine
690 ldrb dir,rb
700 ldrb count,tb
710 cmp dir,#1
720 addeq count,count,#1
730 subne count,count,#1
740 cmp count,#12
750 moveq dir,#0
760 cmp count,#0
770 moveq dir,#1
780 mov col,#3
790 .loop
800 add tint,col,count
810 cmp tint,#15
820 rsbgt tint,tint,#30
830 strb col,param
840 mov R0,#16
850 strb R0,param+1
860 mov tint,tint,LSL#4
870 strb tint,param+2
880 strb tint,param+3
890 strb tint,param+4
900 mov R0,#&C
910 adr R1,param
920 swi "OS Word"
930 add col,col,#1

```

```

940 cmp col,#16
950 blt loop
960 strb dir,rb
970 strb count,tb
980 mov pc,R14
990 ]NEXT
1000 ENDPROC
1010 :
1020 DATA "$RISC User$ brings you a shi
ning example of $ARCHIMEDES$ graphics -
All accomplished in Mode 12 "
1030 DATA "A rapidly changing grey-whit
e palette gives the impression of a refl
ective, highly-polished surface
"
1040 DATA " "

```

MEGA WORM

This clever routine by Barry Christie produces a truly hideous giant worm which ripples away from you while still remaining stationary!



```

10 REM >MegaWorm1
20 REM Author Barry W Christie
30 :
40 MODE 12:OFF
50 FOR I=1100 TO 1 STEP -.25
60 GCOL I MOD 16
70 CIRCLE 640+I*SIN(RAD(I)),512+I*SIN
(RAD(I/2)),I/2
80 NEXT
90 REPEAT
100 FOR I=0 TO 15
110 FOR J=1 TO 7
120 COLOUR I+J,J*16,J*16,J*16
130 COLOUR I+16-J,J*16,J*16,J*16
140 NEXT:WAIT:NEXT:UNTIL FALSE

```



A WINDOW ON THE ARCHIMEDES

(Part 2)

Felix Andrew explains how the procedures presented last month may be used in your own applications, and shows how easily these allow you to design a completely new window.

Last month's program contained all the basic procedures to create and display simple windows using the Archimedes WIMP manager. By following the layout of this program and using the procedures and functions contained in it, you should be able to create your own WIMP-based programs. A good way of doing this is to place the procedures in a library file and use the Basic LIBRARY statement to call them up (see RISC User Issue 2). The use of the various procedures and functions is described in more detail below.

The additional code given this month implements a third window which is described later. This code should be appended to last month's program, keeping strictly to the line numbering as given. You will probably find it worth doing this first so that you may then experiment with the revised program as you read through the rest of this month's instalment on windows.

INITIALISATION

The windows environment is initialised by the procedure PROCsetupvars, which also sets the colour palette and allocates space for the parameter blocks which are used to communicate with the WIMP manager. Line 500 initialises the WIMP manager itself, and this must always be done at the start of any windows program. The WIMP manager clears the background to the highest physical colour (in mode 12 colour 15). Line 480 also clears the screen to the same colour but helps to prevent any unsightly colour changes in the interim before the WIMP manager takes effect. Several macro flags are created for use now and later. These will be dealt with later.

CREATING YOUR OWN WINDOWS

The dimensions of each window, its colours and name are all passed to the function FNcreate (line 1620 onwards). This function does take 11 parameters, but they are all quite simple to understand. The first two (pw and pd) define the size of the page or WAE (work area

extent) in terms of its width and depth (or height). The top left hand corner of the WAE is set to the origin (0,0), so that the WAE width is positive (to the right), and its depth is negative (downwards). The WAE can be much larger than the visible area, and even larger than the screen area as demonstrated by the new window.

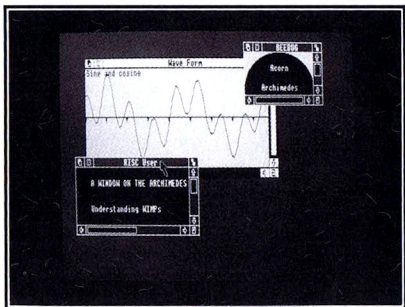
Next we specify the colours for the title bar. You need to specify the foreground colour (tf), background colour (tb) and select colour (ts). In the demonstration program, the mode 12 logical colours are referred to as c1, c2 etc, and the actual colours used are defined in PROCsetupvars (last month's Screen Manager utility was very useful here).

The title of the window is specified next (T\$), and this must contain a maximum of 11 characters. Next we define the foreground and background colours for the WAE (pf, pb), followed by the foreground and background colours of the scroll bars (sf, sb). Finally we need to pass a series of flags, which will be explained in a later article. The WIMP manager gives each window a number or handle which it uses to refer to the window from then on, and this is returned by the FNcreate function. Once the window has been created, the WIMP manager will keep a record of the window's parameters. Now we must open the window. In the demonstration, the additional procedure PROCsetupwindows calls FNcreate to establish all the windows to be used.

OPENING A WINDOW

To open a window (display it on the screen) the WIMP manager needs to know several things. The procedure PROCopenwindow1 (line 2140), with 5 parameters, performs this function. The first parameter is the window handle. Next we specify where on the screen we want the top left hand corner of the window to be. Finally we specify how wide and how high the window should be initially. If you make the window too small, in any direction, for the WIMP manager to be able to draw its frame,

then the window will open at a default minimum size in that direction. If, however, you make the window bigger than the defined size of the WAE, in any direction, then the WIMP manager tries to open it to the size of the WAE, in that direction. Using the routines FNcreate and PROCopenwindow1 you can create and open all the windows you want, but remember that the WIMP manager can only keep track of 32 windows at a time.



FILLING A WINDOW

We now have to put any text or graphics inside the window. If you take a look at the end of last month's program, you will see two short procedures which fill the two windows with text and graphics. Let us go through the first procedure PROCwindow1 (see line 5000). This procedure prints two pieces of text in window 1 using PROCText. The parameters for PROCText are very similar to those used in the Basic instruction PRINTTAB(x,y). First we specify where the text is to appear on the WAE, in graphics units, and then we specify the text itself. To save time, the procedure PROCText will only attempt to print the text if it would currently be visible.

The second procedure, PROCwindow2 from line 5050, performs a similar function for the second window, but contains some additional coding. This procedure draws a circle behind the text. To do this we must first see if any part of the circle is actually visible and needs to be drawn. The circle we want has an origin of 150,-150 on the WAE, and a radius

of 150. The circle therefore occupies a square whose bottom left-hand corner is at (0,-300) and whose top right-hand corner is at (300,0). The procedure FNin (0,-300,300,0) checks to see if any part of this square needs to be drawn. If it does then PROCgwindow is used to set up a normal graphics window (using VDU commands) and origin so that we can plot our circle without worrying about it spilling out of the window and onto the frame. The final VDU26 resets the graphics windows so that the WIMP manager can draw its frame etc.

CREATING A NEW WINDOW

I'll now show, step by step, how to create your own windows, by adding an entirely new window to our working program (see this month's new code).

1. First, create a new window of the required size. Line 191 does this for us and is added to the procedure PROCsetupwindows, where the variable 'w3' is set as its handle.

2. Now we must open the window. Line 125 does this, and is inserted just before the main loop so that the window is open from the start of the program. In a later article I'll show how to use menus to trigger the opening of windows.

3. At this stage the WIMP manager knows about our new window and will prompt us to redraw it when necessary, but we need to amend the program to do this (line 1275).

These are all the additions that need to be made to create a new window. All that remains is to fill the window as required with text and graphics, using a new procedure called PROCwindow3 (line 5200).

Window 3 is going to contain a graph, a horizontal axis and some text. First we display the text as before. The graph is to be drawn to fill the full height of the WAE, so we know that whenever the window moves we will need to draw some new part of the graph. Because of this we don't bother to test first using FNin (as with window 2), to see if any update needs to be made, and can simply set the graphics window and origin. First we draw the axis, line 5270, which is 200 units down in the WAE and stretches all the way across it. Next we add to the axis some markers, lines 5280 to 5320.

A WINDOW ON THE ARCHIMEDES

SCREEN MANAGER UPDATE

These are positioned every 90 units along the axis, and we therefore use the function FNin to see if any of them are in the visible rectangle of the window which we have to fill. Lines 5340 to 5380 draw the graph in the rectangle. The FOR-NEXT loop starts just outside the rectangle, and finishes just beyond it. This is to make sure the graph joins up. Finally we re-set the graphics origin and window.

That's all we have space for this month. I suggest you experiment with the program we have developed so far, and consider adding one or more windows of your own design. I'm sure you'll find it instructive. Next month we will complete our analysis of the basic windows program by dealing with the very important WIMP poll routine, which is what keeps the window displays intact as you move and manipulate the windows on the screen, and we'll also look at some new features as well.

```
125 PROCOpenwindow1(w3,200,900,750,100
0)
191 w3=FNcreate(4000,-400,c2,c1,c3,"Wa
ve Form",c1,c3,c2,c1>windowf)
305 PROCShut(w3)
1275 WHEN w3:PROCwindow3
1530 hp=wtx-wbx:vp=wtY-wby:ebx=xsc+gbx-
wbx
```

```
1540 etx=xsc+gtx-wbx:ety=ysc+gty-wty:eb
y=ysc-gby-wty
1580 VDU 26,5,24,gbx:gby;gtx-2;gty-4;
1590 ORIGIN wbx-xsc,wtY-ysc
5200 DEF PROCwindow3
5210 PROCtext(4,-8,"Sine and cosine")
5220 PROCtext(2000,-8,"less cosine")
5230 PROCtext(3800,-8,"just sine")
5240 PROCgwindow
5250 GCOL 1
5260 D%=90
5270 LINE ebx,-200,etx,-200
5280 IF FNin(ebx,-220,etx,200) THEN
5290 FOR X%=ebx+4-(ebx+4 MOD D%) TO et
x+4-(etx+4) MOD D% STEP D%
5300 MOVE X%-4,-200:MOVE X%+4,-200:PLOT
85,X%,-220
5310 NEXT
5320 ENDIF
5330 GCOL 2:P%=4
5340 FOR X%=ebx-4 TO etx+4 STEP 8
5350 IF X%>ebx-4 THEN P%=5
5360 IF X%<3000 THEN C=COS(RAD(X%/ .27))
*(1-(X%/3000)) ELSE C=0
5370 PLOT P%,X%,(SIN(RAD(X%)))+C)*90-200
5380 NEXT
5390 VDU26
5400 ENDPROC
5410 :
```

RU

SCREEN MANAGER UPDATE

by Lee Calcraft

Here, as promised, are a few additional notes on using last month's Screen Manager.

There was not the space in last month's article to explain how to make use in other programs of the palette files created by the Screen Manager. It is really very simple. Just include in your own program, lines 1960 to 2130 of last month's listing. This will provide you with two procedures: PROCloader and PROCget. To load a palette file, just include the line:

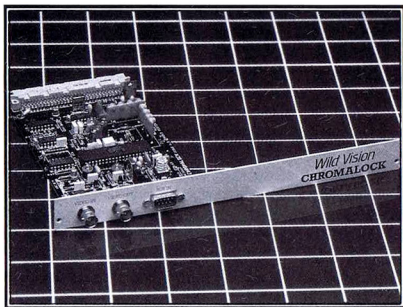
```
PROCloader(filename)
```

where *filename* is the name of the file under which the palette was originally saved. This will set up the full palette of up to 16 colours and the screen border, but since the file contains no mode information, you will need to set the correct screen mode before loading.

One general note on using the Manager itself - the program cannot be fully aware of all the prevailing conditions when it is called, and may therefore need to be tailored in small ways if it is to leave the machine completely unaltered when it passes control back to the calling program. For example, it currently turns on the cursor, and extinguishes the pointer on exit. If this is not appropriate, just alter PROCrestore (lines 710-760). Incidentally, the Screen Manager currently assumes that VDU5 is not in operation - it is safer to add a VDU4 instruction at line 785 at the start of PROCinit, and reinstate VDU5 at line 755 if required.

RU

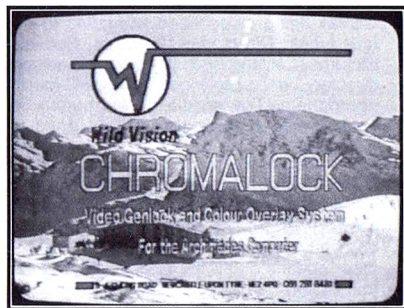
There are many applications which call for a stable PAL encoded video output from the Archimedes, either for directly recording the Arc's output on video tape, or for overlaying its output on a second video signal for titling or other purposes. The G2 Systems Genlock podule opens the way to both of these applications.



It is easily installed in a podule backplane at the rear of the machine, and provides three external sockets: *video in* and *out*, and *RGB in*. The latter connects to the analogue RGB socket of the Archimedes via a short cable supplied with the podule. But the net result of this is that you cannot use the Archimedes' RGB monitor. You can however connect up the mono video output of the Arc to a suitable monitor if you wish. The podule's *video out* could be connected either to a video monitor (mono or colour), or to a video recorder. And finally, the *video in* might either be connected to the output of a video recorder, or to a video camera.

When testing the system, I used a Ferguson Videostar camera as a source, with the output of the podule connected to a video recorder. The system worked first time, producing a combined picture from the Arc, and the camera, as monitored at the output of the video recorder. After a number of tests it was evident that, as suggested in the documentation, the quality of the combined image is not sufficiently good for use with text in 80 column modes. This is a direct consequence of the necessary PAL video encoding performed by the podule,

though on much more expensive broadcast quality Genlock systems the effect is less noticeable. As G2 Systems advise, it is generally better to keep to 40 column text at the smallest; though from tests, it was clear that the degradation was a direct function of the type of background against which the text appeared, and of the colours used in both foreground and background. Certain foreground colours such as deep blue or red also produced a mild patterning, again as a consequence of the PAL encoding. The best titling was obtained using white text of width 20 characters per line or less. Anti-aliased characters also gave reasonable results providing the font size was sufficiently large.



One final point worth noting is that the full Archimedes screen, even including the screen border, is significantly smaller than the standard video frame. Although not a problem for captioning, this immediately precludes the use of this, or any other podule, to overlay the whole screen, as one might wish to do in certain graphics applications. Notwithstanding, if you need a system for overlaying graphics on a video source, then this package may well fit the bill. But you will need to check whether the quality of its output is sufficient for your requirements.

The G2 Genlock podule costs £295.00 +VAT and carriage, and is supplied by:

G2 Systems,
5 Mead Lane,
Farnham,
Surrey GU9 7DY.
Tel. (0252) 712525.



SUPPORT FOR ARTISAN

David Spencer puts Clares' Artisan support disc through its paces.

The Artisan support disc is a utilities disc designed to complement and enhance Clares' Artisan drawing package. Three features are provided: a set of printer drivers, pattern and fade editors, and a rolling display system. The whole package is supplied on a single disc, complete with a 14 page user guide, for £19.95.

When the support disc is booted, it reconfigures the computer to the necessary state and displays six colourful icons at the foot of a white screen. The filing icon brings up a graphical catalogue of the current disc, with the display being categorised according to one of four types: Directories, Patterns, Fades, and Pictures. Clicking on a directory will select that directory, while clicking on a file will load it as appropriate. The middle mouse button brings up a menu allowing the current drive to be changed.

Clicking on the left hand icon brings up a window containing eight options for different printer types. The first option displays the currently loaded picture on the screen; two other options dump the picture in monochrome to a Epson FX or RX printer, two provide colour dumps to Epson colour printers, such as the EX800, two more generate a colour dump on an Integrex 132 printer, and the final option prints the current palette on an Integrex. Clares can supply the high quality paper necessary to get best results from the Integrex dumps. I tried the printer dumps on an RX80 and an FX80, and both produced adequate results. However, as with all dumps, a brand new ribbon is helpful, and the pictures obviously lose a lot of impact when printed just in black and white.

Another icon, in the shape of a monitor, brings up the fade editor. This allows you to determine how, in a sequence of screens, one may fade into the next. For example, the new picture could appear from the centre outwards, or close in from the edges. The fade editor lets you design, on a grid, the process by which the pictures will fade together. You design the fade pattern on a 40 by 32 grid, which represents a

quarter of the fade, the editor mirroring this for the other corners.

The next icon is used to invoke the pattern editor. This allows you to define patterns on an 8 by 8 grid, which can either be used in Artisan for filling pre-defined areas, or to produce half-tones on the printer to simulate colour on a black and white dump.

The final icon, a very colourful rainbow, is used to change the sixteen mode 12 colours used by Artisan. Any colour can be selected and changed by dragging red, green and blue sliders, just as with the Desktop. Unfortunately, the system only maintains one palette, which means that as soon as you redefine any colours, the screen icons and windows change immediately. It would be nice if the software stored a copy of the re-defined palette, and only used this when a picture was displayed.

The other major feature of the Artisan Support disc is the Display module. This is a relocatable module which, when loaded into the computer, provides additional star commands for displaying Artisan screens. A single command can be used to pause for a pre-determined length of time, and then fade in a new mode 12 picture according to a pre-selected fade pattern. Commands also exist to load in a new set of eight fade patterns, and to alter the delay between one picture and the next. A useful feature is the ability to cycle through each mode 12 screen on a disc, using each fade pattern in turn.

In conclusion, this product provides a very useful addition to Artisan. In particular, the printer dumps and the display system are very welcome. I can fully recommend this package.

RU

**Product
Supplier**

**Artisan Support Disc
Clares Micro Supplies
98 Middlewich Road,
Northwich, Cheshire CW9 7DA.
Tel. (0606) 48511
£19.95 inc. VAT**

Price



INTRODUCING ARM ASSEMBLER (2)

by Lee Calcraft

This month: more on the condition mnemonics, implementing FOR-NEXT loops, and a word-search utility.

One of the many powerful features of ARM assembler is that any ARM instruction may be rendered conditional by appending a two-letter condition mnemonic to the instruction. Thus while:

```
ADD R0,R1,R2
```

will add the contents of register R2 to R1 and place the result in R0, the following:

```
ADDCS R0,R1,R2
```

will perform the operation only if the carry flag is set.

The accompanying table gives the complete set of condition mnemonics. It includes two which are largely redundant: AL (always) and NV (never). The first need not be used because if no condition mnemonic is supplied, AL is assumed, and the second is of little use since it simply means that the instruction which it accompanies will never be performed.

AL	Always	VS	Overflow Set
NV	Never	VC	Overflow Clear
EQ	Equal	MI	Minus
NE	Not Equal	PL	Plus
HS	Higher or Same		
or CS	Carry Set	n1>=n2	
LO	Lower		
or CC	Carry Clear	n1<n2	
HI	Higher	n1>n2	
LS	Lower or Same	n1<=n2	
GE	Greater than or Equal	n1>=n2	
LT	Less Than	n1<n2	
GT	Greater Than	n1>n2	
LE	Less Than or Equal	n1<=n2	

Table 1. ARM Condition Codes.

References to n1 and n2 give the effect of the corresponding code after CMP n1, n2.

As you can see, the list has been split into three parts. The codes in the second and third parts are directly equivalent to each other, except that the former apply where unsigned integers are in use, and the latter where the signed integer convention holds. Using this

convention, known as *two's complement representation*, the top bit of all 32 bit words is taken as the sign bit. If this bit is set, the number is treated as a negative one. For further details of two's complement representation, see Cockerell, *ARM Assembly Programming*, p.12.

Thus to branch to the label *addr1* if the value in R0 is less than or equal to that in R1, you could use one of the two pairs of instructions:

```
CMP R0,R1
BLS addr1
```

or:

```
CMP R0,R1
BLE addr1
```

The first pair would work if the registers R0 and R1 contained 32 bit positive integers, while the second pair would be used where two's complement representation was in use.

Although we have said that the condition mnemonics may be used with *any* member of the ARM instruction set, we have only considered their use following the CMP instruction. Three other instructions always affect the processor's flags. These are TST (TeST bits), TEQ (Test EQuivalence) and CMN (CoMpare Negative). More important in many respects is the large group of instructions which can optionally affect the flags. See table 2. The way in which the user indicates that flags should be set by instructions within this group is to add an "S" to the mnemonic. Thus:

```
ADD R0,R1,R2
```

will add the contents of register R2 to R1 and store the result in R0, without altering any of the processor's flags. But the instruction:

```
ADDS R0,R1,R2
```

will perform the addition and set the N, Z, V and C flags according to the result.

The *negative* flag N is set if the top bit of the result is high (indicating a negative result if two's complement representation is in use). The *overflow* flag V is set if there is a signed integer overflow. The *zero* flag Z is set if the



ADC	Add with Carry
ADD	Add without Carry
SBC	Subtract with Carry
SUB	Subtract without Carry
RSC	Reverse Subtract with Carry
RSB	Reverse Subtract without Carry

AND	bitwise AND
BIC	bitwise AND NOT
ORR	bitwise OR
EOR	bitwise EOR

MOV	Move
MVN	Move NOT

Table 2. The Arithmetical and Logical instruction group.

Each member may take an "S" suffix.

result of the addition is zero, and the *carry* flag is set if the addition results in a carry. All the arithmetical instructions in table 2 set these four flags in the same way (providing that the mnemonic contains the "S" suffix). In the case of logical instructions (AND, ORR etc.) N and Z are set according to the result, but the carry flag is set according to the result of supplementary shift operations (to be discussed in a future issue), while the overflow flag remains unaltered.

Thus the simple AND instruction can now take 32 different forms, and might appear in any of the following guises:

```
AND R0,R1,R2
ANDS R0,R1,R2
ANDNE R0,R1,R2
ANDEQS R0,R1,R2    etc.
```

And we have still to consider shifted operands and different addressing modes of this single instruction of the ill-named *reduced instruction set computer*.

CREATING FOR LOOPS

You may be relieved to hear that we have covered sufficient theory for the moment, and can move on to some practical examples. First of all we will implement a simple FOR-NEXT

loop in ARM assembler. This is accomplished in listing 1. If you run this program (patiently) it will display two execution times on the screen, followed by their ratio. The first (around 0.38 seconds) is that taken by the ARM processor to repeatedly assign and multiply two 32 bit numbers, and store the result. It performs this 100,000 times in just 0.38 seconds. A similar operation is then carried out in Basic, and this takes around 40 times as long.

As well as providing a rough indication of the gain in speed of ARM machine code over Basic V, the program also illustrates the use of the "S" suffix in program loops. The assembler program begins on line 100, where it loads the contents of RAM at *table+8* into register R4. This acts as the program loop counter. Lines 120 to 150 fall within the loop, and constitute the multiplication routine. This simply loads two numbers in from memory, multiplies them, and stores the result back in RAM at line 150. Line 160 is the important one. Each time the loop is executed, it causes 1 to be subtracted from the loop counter in register R4. Since the suffix "S" is used, the flags are set according to the result. Line 170 then causes a branch to the label **loop** until the loop counter contains zero. The routine then terminates with the MOV instruction on line 180, exactly as discussed last month. If you forget the "S" in line 160, the loop will carry on for ever.

Listing 1

```
10 REM >2-1ARMpg4
20 REM Timed Multiply
30 :
40 MODE 0:DIM space 200
50 FOR pass=0 TO 1
60 P%=space
70 [
80 OPT pass*3
90 .start
100 LDR R4,table+8
110 .loop
120 LDR R1,table
130 LDR R2,table+4
140 MUL R0,R1,R2
150 STR R0,table+8
160 SUBS R4,R4,#1
170 BNE loop
180 MOV PC,R14
190 :
200 .table
```




INTRODUCING ARM ASSEMBLER

```
210 EQU00:EQU00:EQU00
220 ]
230 NEXT
240 :
250 !table=5000:table!4=99
260 table!8=100000
270 TIME=0
280 CALL start
290 T1=TIME/100:PRINT 'T1;" secs"
300 :
310 TIME=0
320 FOR A%=1 TO 100000
330 B%=5000:C%=99
340 D%=B%*C%
350 NEXT
360 T2=TIME/100:PRINT T2;" secs"
370 PRINT "Speed factor: ";INT(T2/T1)
```

WORD-SEARCHING

We will further put theory to the test in the implementation of a reasonably useful utility. Its function is to search RAM for a specified word-aligned 32 bit word. When the program is run it requests a start and end address for the search. These should be given in hex, though a null Return will substitute PAGE for the start address and HIMEM for the end address. Finally you should supply the 32 bit word to be searched for. This may be entered in decimal, hex (preceded by "&") or binary (preceded by "%"). The search will then commence, displaying the addresses in hex of all finds. You should note that there will always be at least one find between PAGE and HIMEM. This will be the place in RAM where the search word is stored by the program.

The program works as follows. First, R0 is cleared, since this will be used to indicate whether a find has been made or not. Then registers R1, R2 and R3 are loaded with the start address, end address and search value respectively (a more efficient multiple load instruction will be covered next month). The main program loop then begins at line 140. Line 150 loads register R4 with the word held in RAM at the address given by the contents of R1. We are thus using indirect addressing. That is to say, R4 is loaded not with R1 but with the contents of the RAM pointed to by R1 - hence the square brackets around R1.

The central comparison is made in line 160, and if the test word matches the word in RAM,

then the program branches to **out**, and Basic displays the current contents of R1, giving the address of the find. If the words do not match, R1 is incremented by 4 in line 180. This is the loop counter, and the reason that we increment by 4 and not 1 is that on the Archimedes, RAM is addressed in bytes, but we only want to search on 32 bit word boundaries (searching on non-aligned boundaries is more involved).

Line 190 then compares the current address with the end address supplied by the user, and line 200 branches to **loop** if the end address has not been reached. We have used BLO here because we are using unsigned integers for the loop count (since all addresses are positive), and we must exit if the loop count has been matched or exceeded. If we had used BNE here,

then the loop would go on forever in cases where the end address was not on a word boundary.

Once the end address has been reached, R0 is loaded with 255 to indicate this, and the routine returns control to Basic. The Basic program checks the contents of R0 (returned by the USR function), and if it does not contain 255, the start address is incremented, and the machine code is called again. In this way, the program will display every find in the memory range supplied, rather than stopping at the first.

Listing 2

```
10 REM >2-2ARMpg6
20 REM Word Search Routine
30 :
40 DIM space 200
50 FOR pass=0 TO 1
60 P%=space
70 [
80 OPT pass*3
90 .start
100 MOV R0,#0 ;Clear R0
110 LDR R1,table ;Start addr
120 LDR R2,table+4 ;End addr
130 LDR R3,table+8 ;Value
```



```

140 .loop
150 LDR R4,[R1]      ;Load next
160 CMP R3,R4        ;Compare them
170 BEQ out          ;Match found
180 ADD R1,R1,#4      ;Increment counter
190 CMP R1,R2        ;Count finished?
200 BLO loop         ;If not then loop
210 MOV R0,#255      ;Else indicate fail
220 .out
230 STR R1,table     ;Store address
240 MOV PC,R14       ;Return
250 :
260 .table
270 EQU0D:EQU0D:EQU0D
280 ]
290 NEXT
300 :

```

```

310 REPEAT
320 INPUT"Start address? (at word bou
ndary) "&"A$
330 INPUT"End address? "&"B$
340 INPUT"Search word "&"C$
350 IF A$="" A$=STR$~PAGE
360 IF B$="" B$=STR$~HIMEM
370 !table=EVAL("&"A$)
380 table!4=EVAL("&"B$)
390 table!8=EVAL(C$)
400 REPEAT
410 Z=USR(start)
420 IF Z=0 PRINT"Found at "&"~(!table)
430 !table=4+!table
440 UNTIL Z
450 UNTIL FALSE

```

Next month we will grapple with subroutines and SWI calls.

RU

3D GRAPHICS - Continued from page 9

```

1670 LDRB R0,[R12,#1]
1680 LDR R1,[R10,R0,ASL#2]
1690 MOV R1,R1,ASR#8
1700 LDR R2,[R11,R0,ASL#2]
1710 MOV R2,R2,ASR#8:MOV R0,#5:SWI &45
1720 LDRB R0,[R12,#2]
1730 LDR R1,[R10,R0,ASL#2]
1740 MOV R1,R1,ASR#8
1750 LDR R2,[R11,R0,ASL#2]
1760 MOV R2,R2,ASR#8:MOV R0,#5:SWI &45
1770 LDRB R0,[R12,#3]
1780 LDR R1,[R10,R0,ASL#2]
1790 MOV R1,R1,ASR#8
1800 LDR R2,[R11,R0,ASL#2]
1810 MOV R2,R2,ASR#8:MOV R0,#5:SWI &45
1820 LDRB R0,[R12,#0]
1830 LDR R1,[R10,R0,ASL#2]
1840 MOV R1,R1,ASR#8
1850 LDR R2,[R11,R0,ASL#2]
1860 MOV R2,R2,ASR#8:MOV R0,#5:SWI &45
1870 LDMFD R13!,{R15}
1880 :
1890 .face EQU0D
1900 .faceno EQU0D surfs
1910 .draw
1920 STMFD R13!,{R14}
1930 MOV R0,#0:STR R0,face
1940 .drawlp BL plot:LDR R0,face
1950 ADD R0,R0,#1:STR R0,face
1960 LDR R1,faceno:CMP R0,R1:BCC drawlp
1970 LDMFD R13!,{R15}
1980 :
1990 .recaddr ADR R6,recip:MOV R15,R14
2000 .recip

```

```

2010 ]
2020 NEXT
2030 FOR X%=0 TO 449
2040 I%=X%*4:I%!=sin(RAD(X%))*&100
2050 NEXT
2060 FOR X%=1 TO 12799
2070 I%=recip+X%*4:I%=200000/X%
2080 NEXT
2090 ENDPROC
2100 :
2110 DEFPROCdemo
2120 !zoff=25600
2130 FOR xrot=0 TO -20 STEP -2
2140 PROCrotate:WAIT:CLS:CALL draw
2150 NEXT
2160 FOR yrot=-90 TO 90 STEP 4
2170 PROCrotate:WAIT:CLS:CALL draw
2180 NEXT
2190 I=INKEY(0):IF I=32 THEN ENDPROC
2200 FOR zrot=90 TO 360 STEP 6
2210 PROCrotate:WAIT:CLS:CALL draw
2220 NEXT
2230 I=INKEY(0):IF I=32 THEN ENDPROC
2240 FOR X=0 TO 180 STEP 4
2250 zrot=X+4:yrot=96+X:xrot=-24-X
2260 PROCrotate
2270 WAIT:CLS:CALL draw
2280 NEXT
2290 REPEAT:yrot+=4:PROCrotate
2300 WAIT:CLS:CALL draw
2310 UNTIL INKEY(-99)
2320 yrot=yrot MOD 360
2330 ENDPROC

```

RU

THE DREAD DRAGON DROOM

Chris Drage enthuses over a piece of educational software whose stunning graphics will appeal immensely to children of all ages.

Probably the best known educational adventure program on the BBC B is Droom from Resource. Few educational programs enthrall 7 to 11 year olds equally as does Droom; in fact it has become Resource's best seller to primary schools. The willingness with which children tackle and persevere with the mathematical problems encountered is a real credit to Derek Allen, the author of the program. Now Droom is available for the Archimedes. How does it shape up compared with its '8-bit' predecessor?

Droom is based around a story in which the Princess Arminda has been kidnapped by the Dragon Droom and her true love Prince Henry turned into a frog. The children are asked to rescue the princess. A story book provided sets the scene. There are two interlocking parts to the program: the first is the story-line which is a rich source of inspiration for creative and expressive work. The second is a series of puzzles and problems. It is upon the solutions to these that progress through the adventure depends.

A pleasing feature of Droom is that children are not penalised by their mistakes and are not sent back to the beginning of the adventure to start again. Indeed, they are positively encouraged to try again and to succeed. A supplementary puzzle disc is provided in the package and children are able to practise the type of problems encountered in each chapter of the adventure. These include coordinates, digital roots, Pelmanism, Tower Of Hanoi, pattern sequencing, binary numbers, sliding block puzzles, and polygons.

Not surprisingly, Archimedes Droom surpasses the BBC B version in speed,

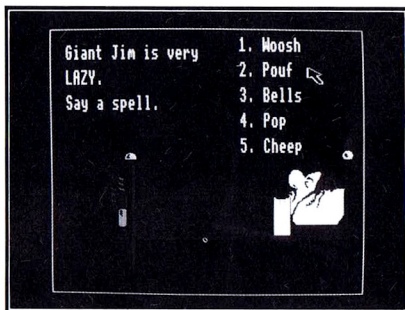
sickness and graphics. Although many children love the mode 6 graphics of the original version, they really go overboard for the stunning mode 13 (256 colour), graphics of the Arc. It is very slick because it is entirely mouse-controlled using the Archimedes' WIMP (windows, icon, mouse, pointer) environment. The animation is very fast and smooth, giving the whole package a very responsive 'feel'. Everything in the BBC version is there in the Archimedes version but in a more sophisticated form. That is not to say that Archimedes Droom

is more complex or difficult. On the contrary a computer-based adventure could hardly have an easier modus operandi.

If I had one small criticism however, it is in the control of the 'car' that children must 'drive' to various locations. On the Arc version children find the 'car' rather awkward to park at each house. A

small point but rather frustrating to those not familiar with the Archimedes version. However, I am pleased to see that in common with Clares Micro Supplies, Resource provide an auto configuration routine on their Droom disc. This automatically resets your machine to its original status on exit from the program.

Droom has enjoyed enormous popularity in schools for two years now. The Archimedes version will appeal to Archimedes owners everywhere who have children. Even at home Droom will provide your youngster(s) with hours of fun and puzzlement. Four colourful screens from Droom are also on this month's disc.



**Program
Supplier**

**Archimedes Droom
Resource
Exeter Road,
Off Coventry Grove,
Doncaster DN2 4PY.
£18.95**

Price

RU

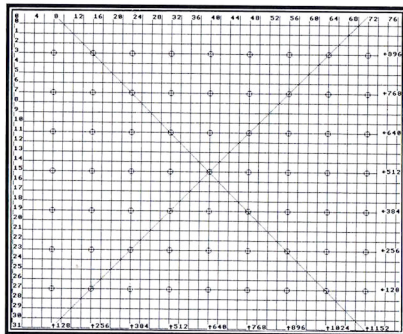
VDU PLANNING SHEET GENERATOR

by Lee Calcraft

This short program will create a screen planning sheet to assist Archimedes graphics design.

The Archimedes can produce some pretty stunning graphics, but in most cases the programmer must carefully plan the screen layout to achieve the best effects. The program presented here uses the Archimedes' built in screen dump to produce a screen planning sheet, complete with graphics grid and character position markings.

To make use of the program, type it in and run it. You should see a grid similar to that in the illustration. If you are happy with it, press the space bar, and it will be dumped out to your printer. The command which performs this task appears in line 120. Alter this to HardcopyMX or RX as appropriate, or to *ScreenSave to save the screen to disc for later dumping.



When you come to use the planning sheet, remember that the graphics area extends right to the edges of the grid. The character positions given are for the common 80 column modes such as 0, 12 and 15 which have 32 lines. To improve the quality of the printout you can run the program in mode 20 if you wish. You will need 160K of screen RAM for this, but you will *not* need a multi-sync monitor. To achieve this, change the MODE statement in line 80 to MODE 20, and reconfigure your machine with

*CON.MON.1 (*CON.MON.0 to reinstate)
followed by Ctrl-Break. After running the program, ignore the broken display, and press the space bar.

```
10 REM >VDUPlan5
20 REM Program VDU Planner
30 REM Version A 0.5
40 REM Author Lee Calcraft
50 REM RISC User May 1988
60 REM Program Subject to Copyright
70 :
80 MODE12:OFF:f=- (MODE=20)
90 *ALPHABET BFONT
100 PROCgrid
110 PROCtext
120 IF GET=32 THEN *HARDCOPYFX 1,1,1,0
,5
130 END
140 :
```

```
150 DEFPROCgrid
160 FOR A=0TO1279 STEP 32
170 MOVE A,0:DRAW A,1023
180 NEXT
190 FOR A=0 TO 1023 STEP 32
200 MOVE 0,A:DRAW 1279,A
210 NEXT
220 LINE 128,1023,1151,0
230 LINE 128,0,1151,1023
240 FOR A=1 TO 9
250 FOR B=1 TO 7
260 CIRCLE A*128,B*128,8
270 NEXT:NEXT:ENDPROC
280 :
290 DEFPROCtext
300 FOR A=0 TO 31
310 PRINTTAB(0,A+(A+1)*f);A;
320 NEXT
330 FOR A=0 TO 39 STEP 2
340 PRINTTAB(A*2,0);2*A;
350 NEXT
360 FOR A=1 TO 9
370 PRINTTAB(8*A,31+32*f);CHR$139;128*
A;
380 NEXT
390 VDU5:GCOL 7,0
400 FOR Y=1 TO 7
410 MOVE 1188,128*Y+12
420 GCOL 0,0:PRINT STRING$(6,CHR$255)
430 MOVE 1200,128*Y+12
440 GCOL 0,7:PRINTCHR$136;128*Y
450 NEXT:VDU4:ENDPROC
```

RU

ADVERTISING IN RISC USER

RISC User has the largest circulation of any magazine devoted to the Archimedes. In fact, we believe over 50% of Archimedes owners are regular readers.

RISC User is therefore the ideal medium for advertising all products for the Archimedes to a discerning and committed readership. With six issues of RISC User successfully complete we can now offer advertising space at attractive rates.

To find out more, contact

Yolanda Turuelo
on (0727) 40303

or write to

**RISC User, Dolphin Place,
Holywell Hill, St Albans,
Herts AL1 1EX.**

MICRO STUDIO

ARCHIMEDES GRAPHIC LIBRARY

THREE DISK SET

£21.95 inc VAT and p&p

BEEB TO ARCHIMEDES DATA LEAD

INCLUDING SOFTWARE

£14.95 inc VAT and p&p

5 1/4 40/80 DISC DRIVE COMPLETE WITH

ARCHIMEDES INTERFACE

£129.95 inc VAT and p&p

*You may pay by Access and Visa,
Educational Orders Welcome.*

MICRO STUDIO(R)

**83 Clay Street, Soham,
Nr Ely, Cambs, CB7 5HL.**

RISC USER Magazine Disc May 1988

The monthly magazine disc contains all the programs from the magazine together with additional items from a variety of sources. Highlights this month include:

FAST DISC COPIER

This utility is particularly useful on single drive systems for fast and efficient copying of files and directories, and also offers substantial improvements on dual drive systems as well.

3D GRAPHICS

A part machine code program demonstrating the principles of hidden-line removal when displaying 3D objects.

WINDOW ON THE ARCHIMEDES

A complete working program incorporating this month's additional window.

INTELLIGENT AUTO- CONFIGURE

Use this utility to take all the worries out of re-configuring your Archimedes, and it can restore the original settings as well.

**All this plus two more
delightful Visuals and two
complete demo programs
from our series on ARM
assembler.**

MODULE FOR TWIN

Install this module on your system for fast access to Twin at all times.

ADDITIONAL ITEMS :

◆ **SERIAL PORT MODULE** - The latest version of this bug-fix from Acorn.

◆ **3D ANIMATION** - All the remaining screen and driving software to install this stunning animation from Acorn (requires last month's disc as well). Full instructions are included.

◆ **DROOM** - Resource, the suppliers of Droom reviewed in this issue, have kindly supplied us with four demo screens for inclusion on this disc. In our opinion, the graphics are quite outstanding.

U-CONNECT

Communications Software for Acorn Archimedes computers

U-Connect is a general-purpose, but full-featured communications software package written specifically for the Archimedes range of computers, using a combination of assembler and C. U-Connect runs exclusively within the Archimedes WIMP environment, fully utilising the concepts of multiple windows, icons, and pop-up menus to provide an easy-to-use, flexible, and attractive package.

U-Connect provides:

Terminal Emulations - Teletype, VT52, VT100 and VideoTex (Prestel)

File Transfers - ASCII, X-Modem, Kermit, CET (Prestel Download)

Screen Review - view past text/frames in separate scrolling window

Modem Drivers - support for all common types of modems, fully user configurable (Hayes, Dacom, Pulsed, and Dumb supplied as standard)

PhoneBooks - Multiple phonebooks allowed, each with 40 entries, and defining phone numbers, logon, function keys etc., allowing the complete environment to be pre-configured automatically before connection.

U-Connect is ideal for the Archimedes User who wishes to connect to remote computer services, such as Bulletin Boards, Online Databases, Conference Systems, E-Mail systems etc. It is supplied on 3.5" disk, with a comprehensive user manual covering all aspects of the package. It may be installed onto hard disk - a Network version will be available shortly.

U-Connect is the first in a range of Magenta Research software packages planned by for the Archimedes over the coming months.

Order: £59.95 inc VAT - p & p within UK (- £4 p & p overseas) Send cheque or PO to:



MAGENTA RESEARCH LTD

5th Floor, Amp House, Dingwall Road, Croydon, CR0 9XA, United Kingdom
Phone: 01-680 2585 International +441 680 2585 Telex: 834302 MAGSYS G

Archimedes

5.25 Disk Interface.

Fit a 5.25 Drive to your Machine the safe and easy way using our
NEW DISK INTERFACE

- ◆ Available Now
- ◆ No Soldering
- ◆ Supports 4 Drives
- ◆ Low Power Consumption
- ◆ Fully Buffered
- ◆ Easy to Fit
- ◆ All Cables Supplied
- ◆ Full Instructions

In Stock Now and Despatched the same day on receipt of your
Cheque / Postal Order for £27-95 inc P&P.

Dudley Micro Services

30 Hadley Close, Netherton, Dudley, West Mids. DY2 9JX

Tel: (0384) 633142

A MODULE FOR TWIN

by Nic Van Someren

Acorn's extremely useful Twin editor is supplied as a non-relocatable piece of code. This short program will modularise it, with a number of attendant advantages.

The main inconvenience of Twin's non-module status is that every time you call it, it must be loaded from disc. This causes delays while loading takes place, and means that you must always have your Twin disc handy. This short piece of code solves the problem. Just type it in and save it away, and make sure that Twin is present in the Library directory of your disc. When you run the program, it will create a relocatable module on disc called **TwinMod**. Now in your Boot file, just place the instruction ***RMLOAD TwinMod**, to load in the module at the start of the day, and whenever you type **TWIN** from Basic or ***TWIN**, Twin will instantly appear. The module works by very rapidly downloading Twin from the RMA whenever it is called (it is not possible to make it run from the RMA). This means that once the module has been created, the original Twin file is no longer needed.

NOTE: You should alter line 80 of the program to give the address from which you wish Twin to run. Remember that Twin takes as workspace the area above its own code. The value given is appropriate if Basic is allocated around 300K of workspace.

```

10 REM >MkTwinMod7
20 REM Program Twin Module Maker
30 REM Version A 0.7
40 REM Author Nic Van Someren
50 REM RISC User May 1988
60 REM Program Subject to Copyright
70 :
80 position%=&24000
90 hand%=OPENIN"%Twin"
100 IF hand%=0 ERROR 23113,"There must
be a copy of Twin in the library"
110 twinsize%=EXT#hand%
120 CLOSE#hand%
130 DIM code% twinsize%+&100
140 FOR pass%=4 TO 7 STEP 3
150 P%&0:0%=code%
160 [OPT pass%
170 EQU D entry ;Application entry
180 EQU D 0:EQU D 0:EQU D 0
190 EQU D title
200 EQU D helpstring
210 EQU D helptable
220 EQU D 0:EQU D 0:EQU D 0:EQU D 0

```

```

230 .title
240 EQU S "TwinInAModule"
250 EQU B 0
260 .helpstring
270 EQU S "Twin Module"
280 EQU B 9
290 EQU S "1.00 (15 Mar 1988)"
300 EQU B 0
310 .twinhelp
320 EQU S "**Twin calls Twin"
330 EQU B 13:EQU B 10
340 .twinsyntax
350 EQU S "Syntax : *Twin [<filename>[
<filename>]]"
360 EQU B 0
370 ALIGN
380 .helptable ;One command only
390 EQU S "Twin"
400 EQU B 0
410 ALIGN
420 EQU D preentry ;Code for command
430 EQU D &00050000 ;0 to 5 params
440 EQU D twinsyntax
450 EQU D twinhelp
460 EQU D 0
470 .preentry
480 MOV R2,R0
490 ADR R1,title
500 MOV R0,#2
510 SWI "OS_Module"
520 .entry
530 LDR R8,thesize
540 ADR R9,theocode
550 ADD R8,R8,R9
560 MOV R10,#position%
570 .moveloop ;Loop to move code
580 LDMIA R9!,{R0-R7} ;Read 8 words
590 STMIA R10!,{R0-R7} ;Write 8 words
600 CMP R9,R8 ;Test for end
610 BCC moveloop
620 MOV PC,#position% ;Enter Twin
630
640 .thesize
650 EQU D twinsize%
660 .theocode
670 ]
680 NEXT
690 OSCLI"LOAD %.TWIN "+STR$~0%
700 OSCLI"SAVE TwinMod "+STR$~code%+"
"+STR$~(twinsize%+theocode)
710 OSCLI"SetType TwinMod FFA"

```

RU

HINTS & TIPS HINTS & TIPS

ZARCH CHEAT MODE

R. Johnson

If you are a Zarch addict, but still find the game a bit too taxing, you might be interested to learn that there is a cheat mode. Once you have started the game, but before taking off, ensure that the sound is turned on and press 'Q', and then very quickly press 'T' and 'U' simultaneously. The game can then be played normally, except that pressing 'D' will select, or deselect, the automatic pilot, while pressing 'L' gives you an extra life, and the 'F' key refuels your tanks instantly.

BEEB STYLE BREAK

Crosbie Fitch

It would sometimes be nice for the Break key on the Archimedes to behave exactly as it does on the other BBC computers; in other words the computer should be reset when Break is pressed on its own. This is easily accomplished by typing *FX247 which makes the Break key generate a reset whenever it is pressed, regardless of whether Shift or Control are pressed at the same time.

BRAINSOFT DISC CONVERSION

James Farlane

In the review of BrainSoft's Disc Conversion package in RISC User Vol.1 Issue 3 it was stated that it does not work correctly with the new Arthur 1.2 operating system. However, the current version of the software will work as long as Acorn's module fixing the serial bug is installed first. This module was included on the disc for RISC User Vol.1 Issue 4, and an improved version is included on this month's disc. Alternatively, the module can be obtained from Customer Services at Acorn in exchange for a blank disc.

DELETE KEY

Crosbie Fitch

This short but clever routine makes the backspace key generate a delete character - very useful if you normally use a computer where backspace is used for delete. The way in which the program works is also quite interesting. When run, the Basic program assembles a short piece of machine code and saves this to disc under the name BStoDEL. This machine code is given the file type

transient which means that if it is *RUN, it will load and execute in the RMA workspace. When executed, the code claims some more space in the RMA area, copies a short routine into this space, points the buffer insert vector to this routine, and then exits. Now, each time a character is entered into the keyboard buffer, the new routine checks to see if it is a backspace (ASCII code 8), and if it is, translates it to a delete (ASCII code 127).

Once the code is assembled and saved, it can be installed at any time with *BStoDEL, and will remain in the machine until it is reset. This routine has the effect of preventing Control-H from producing a backspace, but if desired, the command *KEY13|H can be used to make the Insert key generate a backspace instead.

```
10 DIM code% 100
20 SP=13:LINK=14
30 FOR pass% = 4 TO 7 STEP 3
40 P% = 0: O% = code%
50 [OPT pass%
60 STMFD SP !, {R1-R3}
70 MOV R0, #6: MOV R3, #16
80 SWI "XOS_Module"
90 BVS exit
100 ADR R0, BStoDEL
110 .copy
120 SUBS R3, R3, #4
130 LDRPL R1, [R0, R3]
140 STRPL R1, [R2, R3]
150 BPL copy
160 MOV R0, #&14: MOV R1, R2: MOV R2, #0
170 SWI "XOS_Claim"
180 .exit
190 LDMFD SP !, {R1-R3}
200 MOV PC, LINK
210 :
220 .BStoDEL
230 TEQ R0, #8
240 TEQEQ R1, #0
250 MOVEQ R0, #127
260 MOV PC, LINK
270 ]
280 NEXT
290 SYS "OS_File", 10, "BStoDEL", &FFC, , code%, 0%
```

HINTS & TIPS HINTS & TIPS

CHANGING THE PALETTE

David Spencer

You may not be aware that the VDU 19 command to alter the colour palette can also be used to create new flashing colours, as well as setting the border. The basic syntax of the command is VDU 19,c,p,r,g,b. The value of 'p' defines exactly what the command does.

p=0-15 sets logical colour c to actual colour p, just as on the model B.
p=16 defines logical colour c according to the values of r, g and b, these being in the range 0-255.
p=17 sets just the first flashing colour for colour c, according to r, g and b.
p=18 is the same as above, but sets the second flash colour.
p=24 sets the border colour to r, g and b, with c being ignored.
p=25 changes the pointer colour. The value of c, in the range 1-3, selects the logical colour to be changed, with r, g and b determining the new colour.

For example:

```
VDU 19,1,17,100,0,100,19,1,18,128,128,50
```

will give you a mode 0 foreground that flashes between lilac and pale yellow. Incidentally, changing the palette was covered in more detail on page 24 of RISC User Vol.1 Issue 2.

AUTOMATIC BACKUP

Lee Calcraft

The EXEC file listed below, when saved under the name PB in the Library directory of your disc, will implement a program back-up facility. Each time you type:

```
*PB
```

it will search the Basic program currently in RAM for a filename stored in the first line of the program in the usual format (i.e. REM >Filename). It will then save the file to the directory \$.BACKUP in drive 1, using the name found. If the disc does not contain such a directory, it will create one first.

Because it is not possible to use the ON-ERROR statement in an EXEC file, the file creates a syntax error so as to set ERR=4 at the start (remember that errors do not stop the execution of EXEC files, and the error is invisible because of the VDU21). After the attempted

back-up, the EXEC file checks the error number to see if the BACKUP directory was found. If not, it creates one, and calls itself again to save the program.

```
*| >PB
*| Program Backup
VDU21
Z
P%=PAGE
REPEAT P%+=1:C%=?P%:UNTIL C%=ASC(">") OR
P%>PAGE+20
IF C%<>ASC(">") VDU6:P.TAB(10)**** File
name not found****:VDU7,21:OS. ("FX12
5"):VDU6
*FX15,1
S$=":1.Backup."+$ (P%+1)
VDU6:P.TAB(10)"SAVING ";S$:VDU21
SAVE S$
IF ERR=67798 VDU6:P."Creating new direct
ory":VDU21:OS. ("CDIR:1.$.BACKUP"):OS.
("PB")
VDU6:IF ERR<>4 THEN REPORT:VDU7,13,10
```

SINGLE LINE SAVE

David Spencer & Crosbie Fitch

It is often desirable to save a machine code program or a block of memory (i.e. using *SAVE), and to date stamp it and set a file type at the same time. This can be done as a two stage operation, by first using *SAVE, and then using *SETTYPE to set the file type, and also date stamp the file. There is, however, a better way of doing this on Arthur 1.2, which involves just a single operation. The new method uses the Arthur call OS_File, with an option not implemented on earlier versions of Arthur. The call to use is:

```
SYS "OS_File",10,<file-name>,<file-type>,,<start>,<end>
```

Obviously, <file-name> should be replaced by a string, and <file-type>, <start> and <end> should be specified as numbers (decimal, unless preceded by an &). The 10 specifies the particular OS_File operation which in this case is "Save and date stamp". It is important not to forget the double comma before the start address. For an example of this call in use, take a look at Crosbie Fitch's delete key routine given earlier.

RU

RISC USER magazine

MEMBERSHIP

RISC User is available only on subscription at the rates shown below. Full subscribers to RISC User may also take out a reduced rate subscription to BEEBUG (the magazine for the BBC micro and Master series).

All subscriptions, including overseas, should be in pounds sterling. We will also accept payment by Connect, Access and Visa, and official UK orders are welcome.

RISC USER SUBSCRIPTION RATES

£14.50	1 year (10 issues) UK, BFPO, Ch.I
£20.00	Rest of Europe & Eire
£25.00	Middle East
£27.00	Americas & Africa
£29.00	Elsewhere

RISC USER & BEEBUG

£23.00
£33.00
£40.00
£44.00
£48.00

BACK ISSUES

We intend to maintain stocks of back issues. New subscribers can therefore obtain earlier copies to provide a complete set from Vol.1 Issue 1. Back issues cost £1.20 each. You should also include postage as shown:

Destination	UK, BFPO, Ch.Is	Europe plus Eire	Elsewhere
First Issue	40p	75p	£2
Each subsequent Issue	20p	45p	85p

MAGAZINE DISC

The programs from each issue of RISC User are available on a monthly 3.5" disc. This will be available to order, or you may take out a subscription to ensure that the disc arrives at the same time as the magazine. The first issue (with six programs and animated graphics demo) is at the special low price of £3.75. The disc for each issue contains all the programs from the magazine, together with a number of additional items by way of demonstration, all at the standard rate of £4.75.

MAGAZINE DISC PRICES

	UK	Overseas
Single Issue discs	£ 4.75	£ 4.75
Six months subscription	£25.50	£30.00
Twelve months subscription	£50.00	£56.00

Disc subscriptions include postage, but you should add 50p per disc for individual orders.

All orders, subscriptions and other correspondence should be addressed to:

RISC User, Dolphin Place, Holywell Hill, St Albans, Herts AL1 1EX.
Telephone: St Albans (0727) 40303
(24hrs answerphone service for payment by Connect, Access or Visa card)